Java Card Open Platform
Security Target-LITE

Emission Date	:	18-05-2013
Document Type	:	Technical report
Ref./Version	:	PU-2012-RT-751-v46-1.0-LITE
Number of pages	:	136 (including two cover pages)

LEGAL NOTICE

COPYRIGHT NOTICE

DISCLAIMER OF WARRANTY

Table of Contents

List of Figures

1 Introduction

This chapter identifies this Security Target LITE[1], its TOE, presents its general structure, and introduces the references, notation conventions, and technical terms to be used in the following chapters.

1.1 Identification of this document

Author	Trusted Logic SAS
Address	6 rue de la Verrerie, 92197 Meudon - France
Title	jTOP INFv#46 - Security Target
Version	1.0-LITE
Keywords	Smart Card; Java Card; GlobalPlatform

1.2 Identification of the TOE

Commercial names	jTOP INFv#46 (SLJ 52 Gxx yyy zL)
TOE version	jTOP IFX#v46
IC identifiers	SLE78CLX1600PM-m7820-M11 SLE78CLX800P SLE78CLX360PM

The Infineon Commercial name for this product is: SLJ 52 Gxx yyy zL, where xx may take different values:

- CA: Contact Based (No Mifare)
- LA: Contactless no Mifare
- DA: Dual Interface no Mifare
- LL: Contactless with Mifare
- DL: Dual Interface with Mifare

Where yyy is the NVM size for the Customer (may take following values: 036, 080, 128, 160)

And where z is the Market segment:

- A: ePassport
- B: eDriving License
- C: National eID Open Platform
- D: National eID with Applets

[1] The Complete ST may be sanitized by the removal or paraphrasing of proprietary technical information. This document is named ST-lite. The ST-lite must be a real representation of the complete ST. This means that the ST-lite cannot omit information which is necessary to understand the security properties of the TOE and the scope of the evaluation.

1.3 Revisions and Comments

Version	Issue date	Comments
1.0-LITE	18 May 2013	Final Version

1.4 CC Conformance

This Security Target claims conformance to the following documents defining the ISO/IEC 15408:2005 standard:

- Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, CCMB-2006-09-001, Version 3.1, Revision 3, July 2009.
- Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements, CCMB-2007-09-002, Version 3.1, Revision 3, July 2009.
- Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements, CCMB-2007-09-003, Version 3.1, Revision 3, July 2009.
- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, CCMB-2007-09-004, Version 3.1, Revision 3, July 2009.

Conformance to ISO/IEC 15408:2005 is claimed as follows:

- Part 1: conformant
- Part 2: extended with the FCS_RND.1 family. All the other security requirements have been drawn from the catalogue of requirements in Part 2 of ISO/IEC 15408:2005.
- Part 3: EAL5 augmented with ALC_DVS.2 and AVA_VAN.5 defined in CC part 3.

1.5 PP Claims

This security target is conformant to Java Card Protection Profile [JCSPP], without the Remote Method Invocation (RMI) option, which is not implemented and not included in the evaluation scope. Therefore all the RMI related security entities and requirements from the PP are excluded from this Security Target.

1.6 Conformance Claim Rationale

This Security Target claims a “Demonstrable” conformance with the Java Card Protection Profile [JCSPP].

The TOE type of the current security target is "Java Card 3.0.4 conformant to Global Platform 2.2.1, implemented on a chip of the SLE78CLX1600PM family” and protection profile TOE type is "smart card platform enabled with Java Card technology". TOE types are compatible since the security target's TOE is a smart card that is enabled with Java Card technology.

The scope of the Embedded Software addressed in this Security Target has been enlarged with respect to [JCSPP] so as to include the Card Manager. As a consequence, the A.DELETION assumption is not applicable to the environment since it is upheld by the objective O.CARD-MANAGEMENT which controls the access to card management functions such as deletion of applets. This security objective for the current TOE corresponds to security objective for the environment OE.CARD-MANAGEMENT from the Java Card

Protection Profile [JCSPP]. As Java Card specifications do not address the deletion of Executable Files or applet instances, the A.DELETION assumption assumes that this procedure is performed safely. In this Security Target, that assumption is discharged by the threat T.DELETION.

The security problem definition of the current TOE is more restrictive than the security problem definition of the Protection profile.

The following table shows SPDs in addition to the SPDs found in PP [JCSPP].

Threats	*Description*
T.IMPERSONATE T.INVALID-INPUT T.INVALID-ORDER T.FORCED-RESET T.REPLAY T.BRUTE-FORCE T.LIFE-CYCLE T.RECEIPT	Additional threats to the security problem definition concerning card management, which are compatible with threats of the protection profile but they are more focus to generic attacks on the smartcard platform and Global platform.
Security Objectives	*Description*
O.REQUEST O.INFO-ORIGIN O.INFO-CONFIDENTIALITY O.NO-KEY-REUSE O.INFO-INTEGRITY O.CARD-MANAGEMENT O.RECEIPT O.LIFE-CYCLE O.GLOBAL-CVM O.CVM-BLOCK O.ERROR-COUNTERS O.RECOVERY O.LOCK	Additional security objectives those are relative to card management and Global Platform specifications. These security objectives cover the additional components of the security problem definition. Therefore the security objectives of the protection profile are compatible with the security objectives for the TOE
Organisational Security Policies	*Description*

OSP.FILE-ORIGIN OSP.SECRETS OSP.KEY-LENGTH OSP.NO-RMI-APPLETS OSP.PERSONALIZATION	Additional organizational security policies to the security problem definition, which are compatible with organizational security policies of the protection profile but they are applicable to GlobalPlatform's specifications.
Assumptions	*Description*
A.NATIVE	The A.NATIVE assumption has been added to the TOE defined in this Security Target, which includes the whole Java Card API, including native methods.

2 Overview

This document is the Security Target LITE of the Java Trusted Open Platform (jTOP), an open smart card enabled with the possibility of enlarging and restricting its set of installed on-card applications, either in the pre-issuance and/or the post-issuance phase of its life-cycle. The platform is compliant with both Java Card 3.0.4 and GlobalPlatform Card 2.2.1 specifications. In particular, it implements the GlobalPlatform ID Configuration 1.0. This platform also provides APIs for LDS file system and PACE protocol to facilitate the development and certification of ID applications.

One of the main security goals of jTOP is to counter the unauthorized disclosure or modification of the code and data of the application instances installed on the card, including the application's code, keys and PINs. In order to achieve these goals, jTOP provides the following key security features:

- Logical separation of the data used by different applications (Java Card firewall)
- Runtime monitoring of applet execution (Defensive Virtual Machine)
- Verification of the origin and integrity of Executable Load Files prior to installation.

In addition to this, jTOP provides basic security services to the applications such as:

- Management of cryptographic keys and PINs
- Symmetric and asymmetric cryptography
- Atomic transactions
- Secure communication channels with the terminal

This document has been conceived to prepare a Common Criteria evaluation using the "compositional approach" described in [ETR] and detailed in [ETRSC] for the smart cards. This approach consists in starting from an integrated circuit that has been independently certified by the Chip Manufacturer, and performing an evaluation of the product resulting from embedding a piece of software into it, which make use of some of the results issued from the evaluation of the chip. The integrated circuit has been evaluated according to the [ICPP] Protection Profile.

2.1 Typographic Conventions

A "T", like in T.INSTALL, prefixes the name of the threats. Similarly, an "O" prefixes the security objectives, the string "OSP" prefixes the organizational security policies, the letter "A" prefixes the assumptions and the letter "D" prefixes the assets. The instances of the security functional requirements in [CC2] are identified by the name of the instantiated component, followed by a suffix, like in FDP_ACC.1-FIREWALL.

The TOE being composite, this security target also contains threats, assumptions, organizational security policies, security objectives, security functional requirements and TOE security functions of the chip security target [ICST]. Only those items that directly or indirectly relate to the security issues addressed in [JCSPP] have been selected and reported in this document. The suffix "IC" is used for the security functional requirements coming from that Security Target.

2.2 Associated Documents

2.2.1 Reference Documents

The following documents are cited in this document.

[AIS34] Evaluation Methodology for CC Assurance Class for EAL5+, AIS34, version 1.0, Bundesamt für Sicherheit in der Informationstechnik

[ANSSI] Référentiel Général de Sécurité, Annexe B1 : Mécanismes cryptographiques – Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques de niveau de robustesse “standard”. ANSSI, version 1.20, 26 janvier 2010

[BAC] BSI-PP-0055 – Common Criteria Protection Profile Machine Readable Travel Document with ICAO Application, Basic Access Control – BSI - version 1.10, 25 March 2009

[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1. Revision 3. July 2009. CCMB-2009-07-001

[CC2] Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1. Revision 3. July 2009. CCMB-2009-07-002

[CC3] Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements. Version 3.1. Revision 3. July 2009. CCMB-2009-07-003

[CEM] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1. Revision 3. July 2009. CEM-2009-07-004.

[CPESC] Composite product evaluation for Smart Cards and similar devices, Version 1.0, Revision 1, CCDB-2007-09-001, September 2007

[EAC] BSI-PP-0056 – Common Criteria Protection Profile Machine Readable Travel Document with ICAO Application, Extended Access Control – version 1.10, 25 March 2009

[ETR] ETR-lite for composition, Version 1.1, July 2002. Available at the address www.ssi.gouv.fr.

[ETRSC] ETR-lite for composition, Annex A, Composite Smart Card Evaluation: Recommended Best Practice, Version 1.2, March 2002. Available at the address www.ssi.gouv.fr.

[GPCS] GlobalPlatform 2.2.1 Card Specification (January 2011), including Mapping Guidelines v1.0.1 – Implementation for mapping a GlobalPlatform card based on Card Specification 2.1.1 to a GlobalPlatform card compliant with Card Specification v2.2.1 (January 2011)

[GPCS-A] Confidential Card Content management – GlobalPlatform Card Specification v2.2 – Amendment A v1.0.1, January 2011

[GPCS-API] Java Card API and Export File for Card Specification v1.5, January 2011

[GPCS-D] Secure Channel Protocol 03 – GlobalPlatform Card Specification v2.2 – Amendment D v1.1, September 2009

[AIS34] Evaluation Methodology for CC Assurance Class for EAL5+, AIS34, version 1.0, Bundesamt für Sicherheit in der Informationstechnik

[GPCS-E] Security Upgrade for Card Content management – GlobalPlatform Card Specification v2.2 – Amendment E version 1.0, November 2011

[GPCS-ID] GlobalPlatform Card ID Configuration, ref GPC_GUI_039, version 1.0, December 2011

[ICAO Doc] ICAO Doc 9303, Machine Readable Travel Documents, part 1 – Machine Readable Passports, Sixth Edition, 2006, International Civil Aviation Organization, normative appendix 5

[ICAO TR] ICAO TR Supplemental Access Control for Machine Readable Travel Documents version 1.00, march 23, 2010 (PACE)

[ICPP] Security IC Platform Protection Profile, Version 1.0, June 2007, registered at the BSI under the reference BSI-PP-0035

[ICST] Security Target M7820 M11 including optional Software Libraries RSA – EC – SHA-2 – Toolbox version 1.6, 2011-04-18, Infineon Technologies AG

[JCAPI] Java Card 3.0.4 Application Programming Interface, Sun Microsystems

[JCRE] Java Card 3.0.4 Runtime Environment Specification, Sun Microsystems

[JCSPP] Java Card System Open Configuration Protection Profile, version 3.0, May 2012

[JCVM] Java Card 3.0.4 Virtual Machine Specification, Sun Microsystems

[MRTD] PKI for Machine Readable Travel Documents offering ICC Read-Only Access, International Civil Aviation Organization (ICAO). Version 1.1, October 1st 2004

[PP0035] Security IC Platform Protection Profile, version 1.0, 15 June 2007

[PPUSIM] Java Card Platform Protection Profile Basic Configurations, PU-2009-RT-79, June 17th 2010, version 2.0.2

[PROFILE] jTOP INFv#46 Profile Specification CP-2007-RT-246-46-1.1

[SAC] Protection Profile Machine Readable Travel Document using Standard Inspection Procedure with PACE, version 1.0 – November 2nd 2011

[SSCD] Application Interface for smart cards used as Secure Signature Creation Devices, European Committee for Standardization (CEN), CWA 14890-1:2004 (E), 22nd December 2003

[TR03110] Advanced Security Mechanisms for Machine Readable Travel Documents – Extended Access Control, BSI-TR-03110

2.2.2 Related Documents

The following Trusted Logic's technical reports describe the assurance measures of the TOE:

[CMC] jTOP v46 – Configuration Management Plan, CP-2012-RT-356-1.0.

[PRE] jTOP v46 – Preparative Procedure, CP-2011-RT-731-1.0.

[ATE] jTOP v46 – Test Documentation, CP-2012-RT-347-1.0 to CP-2012-RT-349-1.0..

[DEL] jTOP v46 – Delivery and Operation, CP-2007-RT-015.

[DEV] SSTP– Development Security, CP-2012-RT-353-1.0.

[CMC] jTOP v46 – Configuration Management Plan, CP-2012-RT-356-1.0.

[FSP] jTOP v46 – Functional Specification,CP-2012-RT-348-1.0.

[IGS] jTOP v46 – Card Initialization Phase, CP-2003-RT-52-27-1.9-SERMA-v46.

[LCD] jTOP v46 – Software Life Cycle, CP-2012-RT-355-1.0.

[TDS] jTOP v46 – TOE Design Specification, CP-2012-RT-349-1.0.

[TAT] jTOP v46 – Tools and Techniques, CP-2012-RT-354-1.0.

[OPE] jTOP v46 – Operational User Guidance, CP-2011-RT-732-46-1.3.

[ARC] jTOP v46 – Security Architecture, CP-2012-RT-38-1.0.

[COMP] jTOP v46 – Composite Design Compliance, CP-2011-RT-738-1.0.

[INT] jTOP v46 – Security Function Internals, CP-2011-RT-729-1.0.

2.3 Acronyms

The following acronyms are used in this document:

Acronym	Meaning
AES	Advanced Encryption Standard
AID	Application Identifier
APDU	Application Protocol Data Unit
API	Application Programming Interface
ATR	Answer To Reset
CAD	Card Acceptance Device
CC	Common Criteria
CCM	Card Content Management
CLA	Instruction class (of an APDU command)
CPLC	Card Production Life Cycle Data
CVM	Cardholder Verification Method
DAP	Data Authentication Pattern
DES	Data Encryption Standard
DEMA	Differential ElectroMagnetic Attack
DFA	Differential Fault Analysis
DPA	Differential Power Analysis
DV	Document Verifier
ECDH	Elliptic Curve Diffie Hellman
EEPROM	Electrically Erasable Programmable Read Only Memory
EMA	Electro-Magnetic Analysis
EPA	Emanation Power Analysis
GP	GlobalPlatform
INS	Instruction code (of an APDU command)
IS	Inspection System
ISD	Issuer Security Domain
JAR	Java Archive file
JCAPI	Java Card Application Programming Interface
JCRE	Java Card Runtime Environment
JCSPP	Java Card System Protection Profile
JCVM	Java Card Virtual Machine

Acronym	Meaning
jTOP	Java Trusted Open Platform
LDS FS	LDS File System
MAC	Message Authentication Code
MRTD	Machine Readable Travel Document
OPEN	Open Platform Environment
OS	Operating System
PIN	Personal Identification Number
PKI	Public Key Infrastructure
PP	Protection Profile
ROM	Read Only Memory
RMI	Remote Method Invocation
RSA	Rivest Shamir Adleman
RTE	Run Time Environment
SAR	Security Assurance Requirement
SCP	Smart Card Platform
SCP02	Secure Channel Protocol 02
SD	Security Domain
SF	Security Function
SFR	Security Functional Requirement
SPA	Simple Power Analysis
SSD	Supplementary Security Domain
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functions

2.4 Glossary

Term	Definition
Applet	An application written in Java Card.
Application Code Verification	A static analysis of an Executable Module to determine whether it respects the CAP format and satisfies some essential security properties, such as the absence of pointer arithmetic, uncontrolled control jumps, data-structure overflows, etc..
Application Instance	Instance of an Executable Module after it has been installed and made selectable.
Application Protocol Data Unit (APDU)	Standard communication messaging protocol between a card accepting device and a smart card. See ISO-7816-4.
Application Provider	The institution that owns an Application and is responsible for its behavior.
Application Session	The link between the Application and the external world during a Card Session starting with the Application selection and ending with Application de-selection or termination of the Card Session.

Term	Definition
Asymmetric Cryptography	A cryptographic technique that uses two related transformations, a public transformation (defined by the Public Key component) and a private transformation (defined by the Private Key component); these two key components have a property so that it is computationally infeasible to discover the Private Key, even if the Public Key is known.
Card Administrator	An organization, representative of the Card Issuer, that has control of the smart card's content and life cycle management. During the platform initialization phase, this role is embodied by the Card Enabler. During the platform usage phase, this role is embodied by the Card Issuer or an Application provider owning a SD with card content management privileges. Depending of its privileges, a Card Administrator can lock, unlock or terminate the smart card, download new applets on it, modify the static keys of its SD or retrieve administration information from the smart card. A Card Administrator always acts on behalf of the Card Issuer.
Card Content	Code and Application information (but not Application data) contained in the card that is under the responsibility of the OPEN e.g. Executable Load Files, Application instances, etc.
Card Enabler	The organization responsible for moving a manufactured TOE to the operational state. The person or organization responsible for transmitting the card to the card Administrator.
Card Image Number (CIN)	An identifier for a specific smart card.
Card Issuer	The organization that owns the card and is ultimately responsible for its behavior
Card Manager	Generic term for the card management entities of a GlobalPlatform card i.e. the Open Platform Environment, the Issuer Security Domain, the Supplementary Security Domains.
Card Manufacturer	The organization responsible for integrating the IC containing the embedded software into its carrier, in accordance with the Card Issuer's requirements, to produce a complete card ready for delivery to the Card Enabler.
Card Production Life Cycle Data	A record that uniquely identifies the smart card and the actors involved in its manufacturing and personalization.
Card Session	The period of time during which the card receives power supply from the terminal without receiving a card reset signal.

Term	Definition
Card Unique Data	Data that uniquely identifies a card, made of the Card Image Number and a code identifying the Card Issuer.
Cardholder	The end user of the smart card.
Cardholder Verification Method (CVM)	A method to ensure that the person presenting the card is the person to whom the card was issued.
Chip Manufacturer	The organization responsible for embedding the software of the OS, RTE and GP in the IC ("masking process").
Closed Mode	A mode in which the card restricts card content management operations. When the card is in the Closed Mode it rejects loading more Executable Load Files. There are two possible closed modes: Java Card Static and Native Card.
Controlling Authority	A Controlling Authority has the privilege to keep the control over the Card Content through the mandating of DAP Verification
Embedded Software	The piece of executable code that is masked on the ROM and written in the EEPROM memories of the integrated circuit. It comprises the Operating System, the Runtime Environment, the Card Manager and the bytecode of the installed Java Card Packages.
Executable File	Actual on-card container of one or more Executable Modules. It may reside in immutable persistent memory or may be created in mutable persistent memory as the resulting image of an Executable Load File.
Executable Load File	An Executable File that is in transit to the smart card.
Executable Module	The on-card executable code of a single Application present within an Executable Load File.
Export File	A binary representation of the type and access modifiers of an Executable File in the CAP format. If *B* is a CAP file that imports methods or fields of a CAP file *A*, then the Export File of *A* contains all the information required to perform the bytecode verification of *B*.
GlobalPlatform Registry	A container of information related to Card Content management.
Host	The back end system that supports the smart card. Hosts perform functions such as authorization and authentication, card administration, download of post-issuance Application code and data and transactional processing

Term	Definition
Initialization Data	Any data supplied by the Platform Developer that is injected into the non-volatile memory of the IC by the IC Manufacturer. These data are for instance used for initializing the platform, and to enforce traceability and secure shipment between phases.
Issuer Security Domain	On-card entity providing support for the control, security, and communication requirements of the Card Issuer
Issuing State or Organization	The state that provides the MRTD for the user. This role is a particular case of “Card Issuer”, as it concerns LDS FS API.
Java Card Platform	A collective name for all the components of the Embedded Software (OS, RTE and GP) that transform the IC into a Java Card enabled smart card.
Java Card Static	A closed mode in which no more Executable Load Files may be loaded on the card.
Java Card System	The term used in [JCSPP] to refer to the Runtime Environment, plus those parts of the Card Manager corresponding to the Installer and the Applet Deletion Manager.
Masking Process	The process of embedding the binary code of the Operating System, the Runtime Environment, the Card Manager and a collection of applets into the IC chip.
Message Authentication Code (MAC)	A symmetric cryptographic transformation of data that provides data origin authentication and data integrity.
Mutable Persistent Memory	Memory that can be modified.
Native Card Mode	A closed mode in which the card behaves as a native card. GlobalPlatform commands are rejected when the card is in this mode.
Object	An entity on which a Security Policy is enforced.
Open Platform Environment (OPEN)	The on-card piece of software that manages the GlobalPlatform Registry.
Platform Developer	The organization responsible for developing the code of the basic OS, RTE and GP software.
Platform Personalization Data	Any data supplied relative to the Card Issuer that is injected into the non-volatile memory of the smart card by the Card Enabler. These data are for instance used to personalize the platform with the Card Issuer’s keys, for traceability purposes, and to secure shipment between phases.
Post-Issuance	Phase following the card being issued to the Cardholder.
Pre-Issuance	Phase prior to the card being issued to the Cardholder.

Term	Definition
Private Key	The private component of an asymmetric key pair.
Public Key	The public component of an asymmetric key pair.
Retry Counter	A counter, used in conjunction with the Retry Limit, to determine when attempts to present a CVM value shall be prohibited.
Retry Limit	The maximum number of times an invalid CVM value can be presented prior to the CVM handler prohibiting further attempts to present a CVM value.
Secret Key	A private key. In GlobalPlatform specification, this term refers to a key used to generate a Session Keys during the initiation of a Secure Channel.
Secure Channel	A communication mechanism between an off-card entity and a card that provides a level of assurance, to one or both entities.
Secure Channel Session	A session, during an Application Session, starting with the Secure Channel Initiation and ending with a Secure Channel Termination or termination of either the Application Session or Card Session.
Security Attribute	A logical entity used by a Security Policy to determine whether the outcome of a requested operation may succeed.
Security Domain	On-card entity providing support for the control, security, and communication requirements of the Application Provider.
Security Policy	A set of rules that regulate how certain assets are managed, protected and/or distributed.
Session Key	A key whose lifetime is a card session. In GlobalPlatform specifications, this term refers to the key associated to a Secure Channel and which is used for a secure communication session.
Subject	The entity within the Platform (e.g. Issuer Security Domain, RTE) that acts on behalf of a User to perform some operation on an Object within the scope of a Security Policy.
Supplementary Security Domain	Security Domain other than ISD.
Symmetric Cryptography	A cryptographic technique that uses the same secret key for both the originator's and the recipient's transformation.
User	Either an Application (via GP API or JC API) or an off-card entity (via an APDU command) that makes a request to a Subject to perform some operation on an Object within the scope of a Security Policy.

3 TOE Description

This part of the document describes the TOE as an aid to the understanding of its security requirements. It addresses the product type and the general IT features of the TOE.

3.1 The Target of Evaluation

The TOE is a smart card composed of a piece of software embedded into a chip of the SLE78CLX1600PM family which transforms the card into a secure open platform device capable of hosting multiple Java Card applications.

The TOE is compliant with the GlobalPlatform Card Specification 2.2.1 [GPCS] and, more specifically, with the GlobalPlatform Card ID Configuration 1.0 [GPCS-ID]. These specifications define means to operate the card in order to download, install and remove applications, select an application (for execution), manage the life cycle of the card and applications, manage cryptographic keys involved in secure messaging or other management operations, and manage a global PIN shared among all applications.

The Java Card technology combines a subset of the Java programming language with a runtime environment optimized for smart cards and similar small-memory embedded devices [JCVM]. The Java Card platform is a smart card platform enabled with Java Card technology (also called, for short, a “Java card”). This technology allows for multiple applications to run on a single card and provides facilities for secure interoperability of applications. Applications running on the Java Card platform (“Java Card applications”) are called applets.

The TOE is compliant with the version of the Java Card platform specified in [JCVM], [JCRE] and [JCAPI]. It includes the Java Card Virtual Machine (JCVM), the Java Card Runtime Environment (JCRE) and the Java Card Application Programming Interface (JCAPI). The TOE does not implement the Java Card optional Remote Method Invocation (RMI) functionality. As the terminology is sometimes confusing, the term “Java Card System” (JCS) has been introduced in [JCSPP] to designate the set made of the JCRE, the JCVM and the JCAPI. The JCS provides an intermediate layer between the operating system of the card and the applications. This layer allows applications written for one smart card platform enabled with Java Card technology to run on any other such platform.

The JCVM is a bytecode interpreter embedded in the smart card. The JCRE is responsible for card resource management, communication, applet execution, and on-card system and applet security. The JCAPI provides classes and interfaces for the core functionality of a Java Card application. It defines the calling conventions by which an applet may access the JCRE and native services such as, among others, I/O management functions, PIN and cryptographic specific management and the exceptions mechanism. The JCAPI is compatible with formal international standards, such as ISO7816, and industry specific standards, such as EMV (Europay/Master Card/Visa).

The TOE can be configured so that new applets can be downloaded and installed, even after the smart card has been issued to the Cardholder. This allows Card Issuers to dynamically respond to their customers changing needs. For example, if the Card Issuer decides to upgrade some of the applications offered to the customer, he can make this change without issuing a new card. Moreover, applications from different vendors can coexist in a single card, and they can even share information between them. A smart card application, however, is usually intended to store highly sensitive information, so the sharing of that

information must be carefully limited. Applet isolation is achieved through the Java Card Firewall mechanism defined in [JCRE]. That mechanism confines an applet to its own designated memory area, thus each applet is prevented from accessing fields and operations of objects owned by other applets, unless the applet that owns it provides a specific interface for that purpose. This access control policy is enforced at runtime by the JCVM. However, applet isolation cannot entirely be granted by the firewall mechanism if certain well-formedness conditions are not satisfied by the applications loaded on the card. A bytecode verifier can statically verify those conditions.

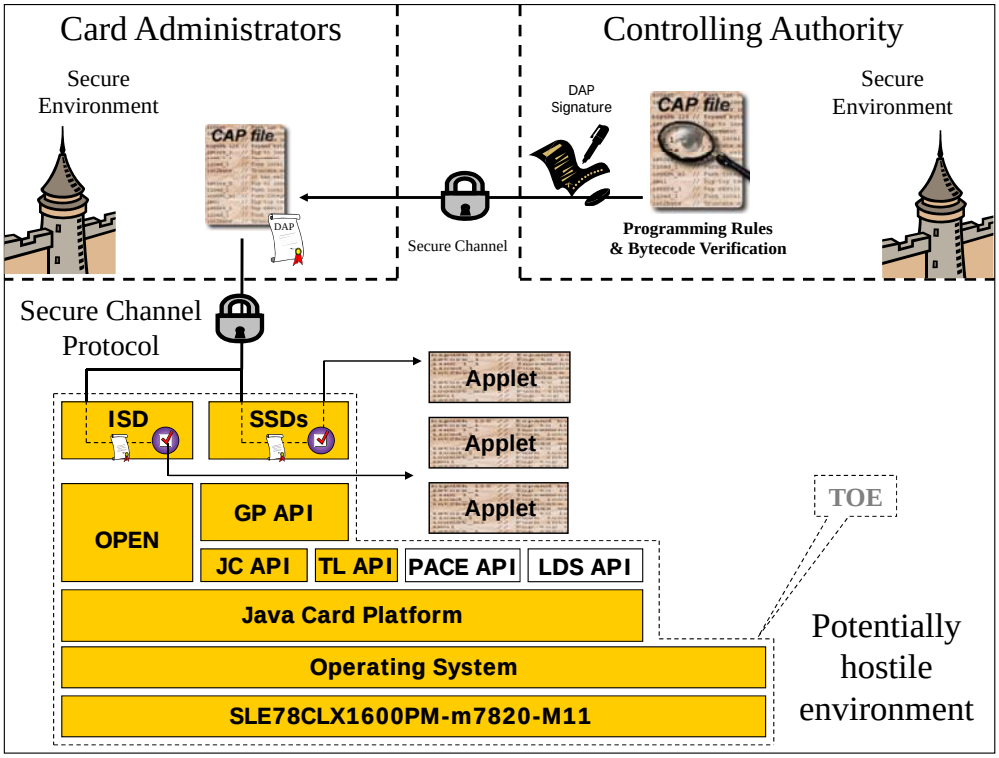

Figure 1: The TOE and its environment

Figure 1 places the different components of the TOE in their environment. Applet source code development is carried on in a Java programming environment. The compilation of that code produces the corresponding class file. This latter file is then processed by the *converter*, which, on the one hand, validates the code, and on the other hand, generates a Converted APplet (CAP) file, the equivalent of a JAR file for the Java Card platform. A CAP file contains an executable binary representation of the classes of a package. A package is a name space within the Java programming language that may contain classes and interfaces. In the context of Java Card technology, it defines either a user library, or one or several applets. This development of the Java source file and its conversion to the CAP format is not relevant for the security of the TOE, and are not included in the figure.

In order to download a new package on the smart card, its code has to be first approved by the Controlling Authority. This Controlling Authority is responsible for checking that the Applet Developer has enforced all the security recommendations that the Platform Developer has stated in the TOE's Operational Guide [OPE], and in particular that the CAP format of the package successfully passes a *bytecode verifier* program. Such verifications are performed in

a secure physical environment that prevents unauthorized people to access to the applet code. If they are successful, the Controlling Authority may electronically sign the CAP file, for instance, using GlobalPlatform's Data Authentication Pattern mechanism (DAP). This signature attests that the Controlling Authority has validated the CAP file, and prevents any further modification on it. The Controlling Authority then transmits the signed CAP file to a representative of the Card Issuer in charge of loading new applets on the card, called hereto a Card Administrator. If the Controlling Authority does not sign the applet, then it is assumed that there is a secure communication channel between the Controlling Authority and the Card Administrator that ensures the origin and the integrity of the received Executable File.

Upon reception of the CAP file, the Card Administrator stores it in its secure environment until the file is downloaded into the card. The Card Administrator transmits the CAP file from its secure environment to the card using a GlobalPlatform's secure channel protocol SCP02. This protocol ensures that the file actually comes from a representative of the Card Issuer and that its integrity has been preserved during the transmission step. The file is received by the Issuer Security Domain (ISD), the on-card representative of the Card Issuer, or by a Supplementary Security Domain with card content management capabilities, the on-card representative of the Application Provider. The Platform Enabler may have configured the card so that the DAP signature of the Controlling Authority is required and verified each time a new CAP file is loaded to the card. In this case, the card may proceed with the loading of the file only upon successful verification of this DAP signature. Once the file has been received and the package it contains has been *linked*, it is possible to install instances of any of the applets defined in the file. During the installation process, every applet is registered on the card by using an Application IDentifier (AID). This AID allows unique identification of the applet instance within the card. In particular, the AID is used for selecting the applet instance for execution. The bytecode interpreter residing on the card, usually called the Java Card Virtual Machine, performs the execution of the applet's code.

The following sections further describe the components involved in the use of a Java Card platform. Although some of these components are not part of the TOE, a better understanding of the role they play will in turn allow the reader to grasp the importance of the assumptions that will appear concerning its environment.

3.1.1 Bytecode Verification

The bytecode verifier is a program that performs a static analysis of the bytecode contained in a CAP file prior to the execution of the file on the card. The actual verifications that the bytecode verifier performs are implementation-dependent, it shall at least enforce all the "must clauses" imposed in [JCVM] on the Java Card bytecodes as well as the correctness conditions of the CAP format described in that document.

Bytecode verification is one of the cornerstones of the security architecture of a Java Card platform. It ensures that the bytecodes contained in the CAP file hold up to their intended use. For instance, it ensures that they do not use or forge fake references to memory blocks, perform illegal control jumps, or use return addresses as if they were integers. In particular, the correct working of the Java Card Firewall depends on some of the properties checked during bytecode verification.

Bytecode verification is performed off-card in the secure environment of the Controlling Authority, and prior to downloading the CAP file on the card. The DAP signature attests that the CAP file has successfully passed bytecode verification.

3.1.2 Installation of Executable Files

Following Java Card specifications, the [JCSPP] introduces the notion of *Applet Installer* as the application of the platform in charge of downloading, linking and installing new CAP files. In a platform compliant with GlobalPlatform Card Specification 2.2.1, Security Domains (SD) with card management privileges play the role of the installer, and CAP files are called Executable Files[2].

When selected, the SD can receive an Executable Load File to be installed on the card. The file is usually sent along several consecutive APDUs. The SD collects all the APDUs, links the whole file to the libraries already installed on the card and initializes the static data, if any.

3.1.2.1 Loading

Loading an Executable File into the card includes two main steps: there is first an authentication step by which the Card Administrator and its Security Domain recognize each other using the SCP02 cryptographic protocol defined by GlobalPlatform. Once the identification step is accomplished, the Executable File is transmitted to the card through some medium that is not supposed to be secure. Due to resource limitations, usually the file is split by the Card Administrator into a list of Application Protocol Data Units (APDUs), which are in turn sent to the card. The receiving Security Domain controls that it has received all the Executable File pieces in the correct order and without modification. If the card has been configured to enforce mandatory DAP verification, then the SD with Mandated DAP Verification privilege verifies the electronic signature that the Controlling Authority attached to the file. In this case, further steps are performed only if the DAP signature is correct.

Loading Executable Files requires the TOE to be configured to support this feature. This feature can be disabled so that the card becomes a *static Java Card Platform*. In this configuration, the platform rejects any attempt of downloading new Executable Files. The set of available applets is the one that can be created from the Java Card packages that have been masked in ROM with the code of the platform and those that have been loaded before moving to the static mode. This operation cannot be undone: once the card becomes static, it cannot rollback to the open configuration again.

3.1.2.2 Linking

The linking process consists of rearranging the information contained in the CAP file in order to speed up the execution of the applications. There is a first step where indirect external and internal references contained in the file are resolved, by replacing those references by direct ones. This is what is elsewhere called the resolution step. In the next step, called the preparation step, the static fields and the statically initialized arrays defined in the CAP file are allocated and initialized. After this step, the CAP file is ready to be executed by the embedded Java Card Virtual Machine.

3.1.3 Installation of Application instances

Each Executable File in the CAP format may contain several Executable Modules, or Applet classes in the Java Card jargon. The actual installation of Applet instances is an independent process that may be delayed in time. Applet installation is then usually separated from the process of loading and linking an Executable File.

[2] The term *Executable File* will be preferred in the sequel to its synonyms *Package* and *CAP file*.

GlobalPlatform defines a specific APDU command for creating an instance of one of those Applet classes. This command specifies the Applet Identifier (AID) to be used for selecting the new Application instance and the privileges to be granted to it. The application in charge of processing installation commands is also the Security Domain owned by the Card Administrator sending the APDU command to the card.

3.1.4 Deletion of Application instances and Executable Files

Executable Files and application instances installed on the card may be deleted on demand of the Card Issuer. Three possible deletion cases are considered in Java Card specifications:

- deletion of an application instance, which is the removal of the applet instance and the Java Card objects created by that instance;
- deletion of a Java Card library, which entails the removal of all the card resident components of the Executable File, including code and any associated management structures;
- deletion of an Executable File declaring applets, which is the removal of the card resident code and JCRE structures associated with the Executable File, as well as of all the instances of the applets that the Executable File declares.

Deletion is only possible when no other Executable File or Application instance depends on the item to be deleted.

In [JCSPP], the *Applet Deletion Manager* is introduced as the on-card component in charge of the mechanisms necessary to delete an applet or a CAP file. According to GlobalPlatform specifications, this role is also embodied by a SD with corresponding privilege.

3.1.5 Card Management

The Card Manager is the on-card component responsible for the administration of the smart card. Its functionality, which goes beyond the scope of the [JCSPP], is also included in the scope of this Security Target.

The Card Manager enforces the security policies of the Card Issuer on the card and provides the following supplementary services:

- Life cycle management of both the whole card and of each of the application instances installed on it.
- Management of logical channels, so that the services of several applet instances may all be active at the same time. Logical channels also enable a given applet instance to be active on different logical channels. The applets that support this latter feature are called *multi-selectable* applets[3].
- Dispatching of the APDU commands to the currently active application instances[4].
- Secure communication channels between the application instances on the card (and specially a SD) and the Card Administrator or the Application Provider.
- A global PIN that can be shared by all the application instances.

According to GlobalPlatform specifications, the Card Management functions described above are shared between the OP Environment (OPEN) and SDs. The services they provide are

[3] This service is explicitly excluded from the evaluation scope of this Security Target; see Section 3.2.1.

[4] In the evaluation scope of this Security Target there can be at most one active applet instance at a time; see Section 3.2.1

available either through APDU commands or through GlobalPlatform's Application Programming Interface.

The set of administration functions supported by jTOP is described in [GPCS-ID]. This latter document describes both mandatory and optional features whose complete description is sometimes to be found in the following documents: [GPCS], [GPCS-API], [GPCS-A], [GPCS-D], [GPCS-E]. We recall below the features described as optional that are actually supported by jTOP (for mandatory features, please refer to [GPCS-ID]):

- Supplementary Security Domains
- SCP02 option '55'
- Delegated Management (DM)
- Token Verification based on RSA1024, AES128 and ECC256
- Receipt Generation based on DES128, AES128
- DAP Verification based on RSA1024, AES128, ECC256

Optionally, the card may be configured to behave like a static Java Card platform. In this case, it is not possible to load new Executable Files anymore. The card may also be configured to behave like a closed native platform. In this case, after installing the desired applet instances, the ISD and all Security Domains become no longer selectable and all their management interfaces become inaccessible. This operation cannot be undone. Once the card enters this mode, the only card management command that is supported is the SELECT command specified in ISO7816. Other APDU commands are directly forwarded to the selected application.

3.1.6 Services to the Applets

In order to enforce an adequate level of security, the platform provides the applets with a collection of frequently used, highly secure services. Those services are available to the applet through Application Programming Interface (API), including:

- Java Card API
- GlobalPlatform API
- LDS FS API (see 3.1.7 for more details)
- PACE API (see 3.1.8 for more details)

The Java Card Application Programming Interface provides the following services:

- Cardholder identification and management of Personal Identification Numbers (PIN);
- Symmetric and asymmetric cryptography services, including encryption and decryption, electronic signature generation and verification, and generation of random data and unique hash values;
- Access to some of the internal runtime data areas of the Java Card Virtual Machine, like determining which is the applet that invokes a given service;
- Controlled sharing of class instances between applet instances;
- Allocation of transient arrays, whose lifetime is either restricted to the card session or to the application sessions with the active applets instance of the same Executable File;
- Management of atomic transactions;
- Abstraction of the low-level communications with the CAD;
- Garbage collection of those class instances and arrays that have become unreachable;

The GlobalPlatform API provides the following services:

- Secure communication with secure channel
- A Cardholder Verification Method consisting of a Global PIN
- Cardholder Verification,
- Personalization
- Card content management services like application loading, card locking or application life cycle update

3.1.7 LDS FS API

The LDS FS API is part of the TOE but is not part of the TSF for this security target. The supported services related to this API are:

- BAC DES Authentication and secure messaging
- Secure Messaging using DES and AES session keys
- LDS File System with fast file reading
- LDS File System with fast file writing for personalization
- Secure storage of biometric and sensitive files.

LDS FS is a file system that contains information like identity, age, name, first name, picture… It may also contain biometric data (fingerprint…). If biometry is present, the EAC configuration is used, and the sensitive files are stocked securely. EAC enables protection of sensitive data. If biometry is not present, the BAC configuration is used. The API allowed to manage the file system (add file, read file, set authentication level) and the secure messaging (wrap, unwrap, authentication).

LDS FS is an API at disposal of applications. Evaluated applications making use of this API must include corresponding TSFs in the scope of their evaluation.

3.1.8 PACE API

The PACE API is part of the TOE but is not part of the TSF for this security target. It provides the following services:

- SAC PACE authentication (with ECDH and DES/AES algorithms)
- Secure Messaging initialization with session keys issued from the PACE authentication
- PACE mapping (point generation with ECDH and domain generation)

PACE API is aimed at providing a replacement for the BAC protocol, correcting the entropy weakness with strong session keys. This API provides services as secure messaging and mapping.

This API can be used by applications in the same way as LDS FS API.

3.1.9 Smart Card Platform: Operating System, Dedicated Software and Chip

In the [JCSPP], the Java Card System lays on a *Smart Card Platform*, which is composed of a micro-controller and an operating system. It provides memory management functions with separate interface to RAM and EEPROM, I/O drivers compliant with ISO standards, a low level transaction mechanism, and secure and highly efficient implementation of cryptographic

functions. It also contains dedicated software, which provides interface with the integrated circuit.

In this Security Target, the Smart Card Platform is composed of an SLE78CLX1600PM chip and an operating system developed on top of it.

3.2 Limits of the TOE

This section specifies the components of the smart card that form the Target of Evaluation, and the phases of its life cycle that fall under the scope of the evaluation.

3.2.1 Scope of Evaluation

The scope of the TOE is the SLE78CLX1600PM chip and jTOP® as a platform for execution of Java Card applications. It includes the Java Card Virtual Machine (JCVM), the Java Card Runtime Environment (JCRE), the Java Card Application Programming Interface (JC API), and the Card Manager (GP 2.2.1 based) allowing the management post-issuance of off-card verified applications. The TOE does not embed native applications.

The Java Card applets are also excluded from the scope of the TOE, because they are considered as data managed by the TOE. This means that any application-specific TSF is out of the scope of this Security Target. Moreover, the requirements in this Security Target do not span (actually, they do not need to span) all the stages in the development cycle of a Java Card application. Applets are only considered in their CAP format, and the process of compiling the source code of an application and converting it into the CAP format does not regard the TOE or its environment. On the other hand, the processes of verifying CAP files and loading them on the card are a crucial part of the TOE environment and play an important role as a complement to some of the on-card security functions. For this reason, this Security Target requires the enforcement of organizational security policies regarding those activities, and imposes security functional requirements on the implementation of the bytecode verifier.

The TOE includes the part of the Java Card Application Programming Interfaces that the platform supports plus some additional (proprietary) Java Card libraries developed by Trusted Logic. These proprietary libraries are:

- `fr.trustedlogic.javacard.internalservices`
- `fr.trustedlogic.javacard.security`

The LDS FS and PACE APIs are supported by the Platform, but are excluded from the scope of evaluation. For this reason, this Security Target does not require the enforcement of organizational security policies regarding those activities, and does not impose security functional requirements. The following additional Java card libraries related to PACE and LDS are also excluded from the scope evaluation:

- `fr.trustedlogic.javacard.lds.filesystem`
- `fr.trustedlogic.javacard.pace`

The following Java Card libraries are also included in the chip mask, but are excluded from the scope of evaluation:

- `fr.trustedlogic.math.modular`
- `fr.trustedlogic.javacard.framework`

The TOE has been designed to support a configurable number of logical channels, which can be set up during the initialization phase of its manufacturing process. For the sake of the evaluation, it is assumed that the Card Manufacturer initializes the TOE so that one single logical channel can be opened at most.

Regarding the TOE's life cycle, this Security Target only covers the development of the software to be embedded on the IC and those states once the TOE has become operational, that is, once the code of the Java Card System can be executed. The construction of the IC and the smart card and embedding process itself are addressed in [ICST] and are out of the scope of this Security Target.

3.2.2 Users and Roles

The users of the TOE include the following people and institutions.

Platform Manufacturer

The Platform Manufacturer is a generic term that includes all the actors involved in the Platform Development phase of the smart card's life cycle. During this phase, the role of the Platform Manufacturer is in turn embodied by the Application Provider, the Platform Developer, the IC Manufacturer, and the Card Manufacturer.

Platform Developer

The Platform Developer is the organization responsible for designing and implementing the software masked on the IC. This includes the following components of the TOE: Card Manager, Java Card Runtime Environment, and Operating System.

Application Provider

An Application Provider is an organization that develops Java Card applications on demand of the Card Issuer. These applications implement services that the Card Issuer proposes to the Cardholder.

Application providers are represented on card by Supplementary Security Domains, as defined in [GPCS]

IC Manufacturer

The IC Manufacturer integrates the Embedded Software within the IC. This is usually known as the "masking" process.

Card Manufacturer

The Card Manufacturer integrates the masked IC with the carrier (a plastic card, a passport booklet, etc) in accordance with the Card Issuer's requirements, to produce a complete smart card ready for delivery to the Card Enabler.

Card Enabler

The Card Enabler is responsible for preparing the smart card for platform initialization according to the instructions of the Card Issuer. It may load configuration data into the smart card and also load or replace the static ISD keys to be used for the authentication of the Card Administrator.

Card Administrator

The Card Administrator is an organization, representative of the Card Issuer, that has control of the smart card's content and life cycle management. During the platform initialization phase, this role is embodied by the Card Enabler. During the platform usage phase, this role is embodied by the Card Issuer or an Application provider owning a SD with card content management privileges. Depending of its privileges, a Card Administrator can lock, unlock or terminate the smart card, download new applets on it, modify the static keys of its SD or retrieve administration information from the smart card. A Card Administrator always acts on behalf of the Card Issuer.

Controlling Authority

The Controlling Authority is in charge of ensuring that the Java Card applets to be installed on the smart card do not violate any of Card Issuer's security policies. In particular, the Controlling Authority is responsible for the bytecode verification of the downloaded applets, as well as for checking that the Application Developer did respect all the security recommendations specified in [OPE].

As described in [PRE], the Platform Enabler may install a Supplementary Security Domain on the card representing the Controlling Authority and responsible for verifying a signature generated by the Controlling Authority for every Java Card package loaded to the card.

Cardholder

The Cardholder is the person or group of persons that the Card Issuer designates as the rightful holder of the smart card.

Card User

A Card User is any person presenting the smart card to the terminal and claiming the identity of the Cardholder.

3.2.3 The TOE in the Smart Card's Life Cycle

The TOE life cycle is part of the product life cycle, i.e. the IC with Java Card platform and applications, which goes from product development to its usage by the final user. The

product life cycle phases are those detailed in Figure 3. We refer to [PP0035] for a thorough description of Phases 1 to 7:

- Phases 1 and 2 compose the product development: Embedded Software (IC Dedicated Software, OS, Java Card System, other platform components such as Card Manager, Applets) and IC development.
- Phase 3 and Phase 4 correspond to IC manufacturing and packaging, respectively. Some IC pre-personalisation steps may occur in Phase 3.
- Phase 5 concerns the embedding of software components within the IC.
- Phase 6 is dedicated to the product personalisation prior final use.
- Phase 7 is the product operational phase.

The Java Card System life cycle is composed of four stages:

- Development,
- Storage, pre-personalisation and testing
- Personalisation and testing
- Final usage.

These stages map to the typical smartcard life cycle phases of [PP0035] as shown in Figure 2.

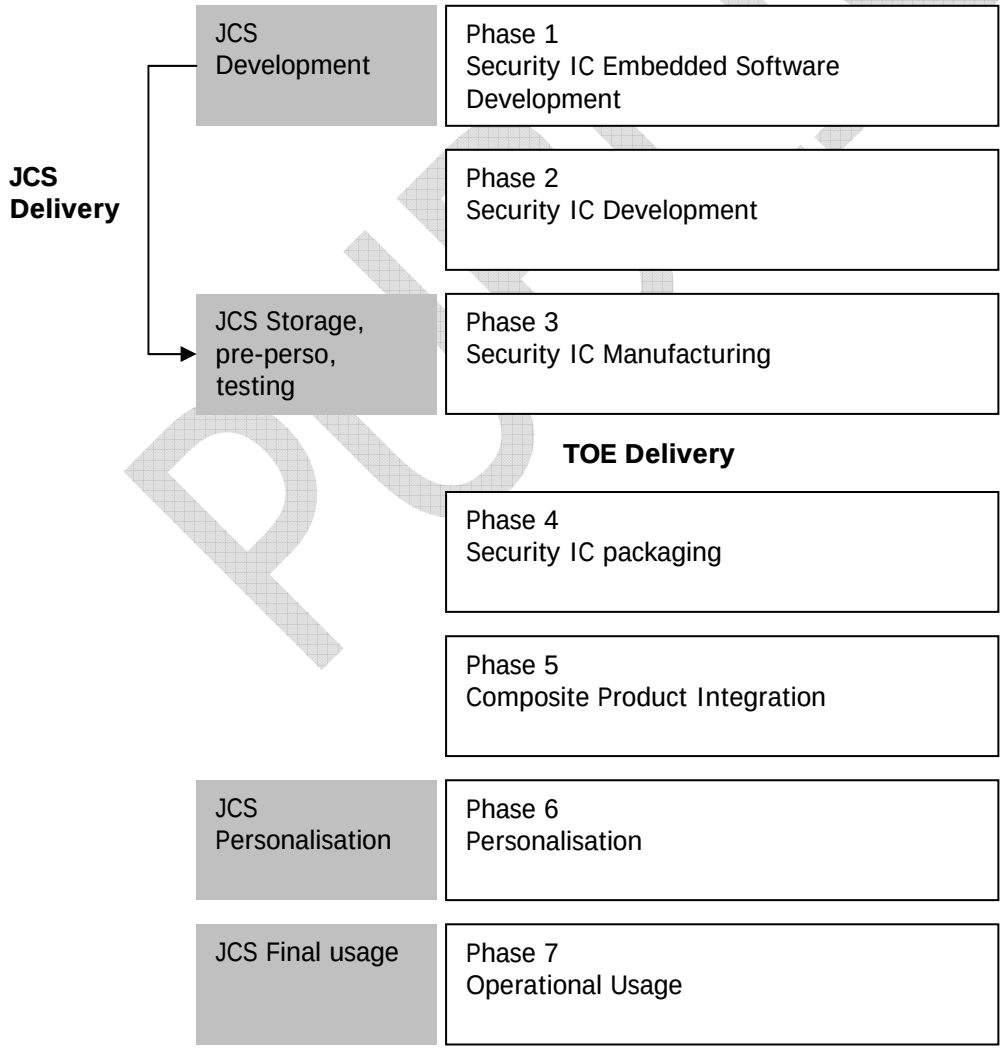

Figure 2: JCS (TOE) Life Cycle within Product Life Cycle

JCS Development is performed during Phase 1. This includes JCS conception, design, implementation, testing and documentation. The JCS development fulfils requirements of the final product, including conformance to Java Card Specifications, and recommendations of the SCP user guidance. The JCS development occurs in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of these elements. The evaluation of the TOE includes the JCS development environment.

In Phase 3, the Security IC Manufacturer stores, pre-personalizes the JCS and potentially conduct tests on behalf of the JCS developer. The Security IC Manufacturing environment protects the integrity and confidentiality of the JCS and of any related material, for instance test suites. The evaluation of the TOE includes the whole Security IC Manufacturing environment, in particular those locations where the JCS is accessible for installation or testing. If the Security IC has already been certified (e.g. against [PP0035]) there is no need to perform the evaluation again.

The **delivery of the TOE** occurs at end of Security IC Manufacturing (Phase 3). Delivery and acceptance procedures guarantee the authenticity, the confidentiality and integrity of the exchanged pieces.

Phases 4, 5 and 6 are done in the Infineon manufacturing sites. They take place in a controlled environment (secure locations, secure procedures and trusted personnel). For the JCS platform preparation and personalization, these phases are covered by Preparation Guidance [PRE] document which is part of the evaluation.

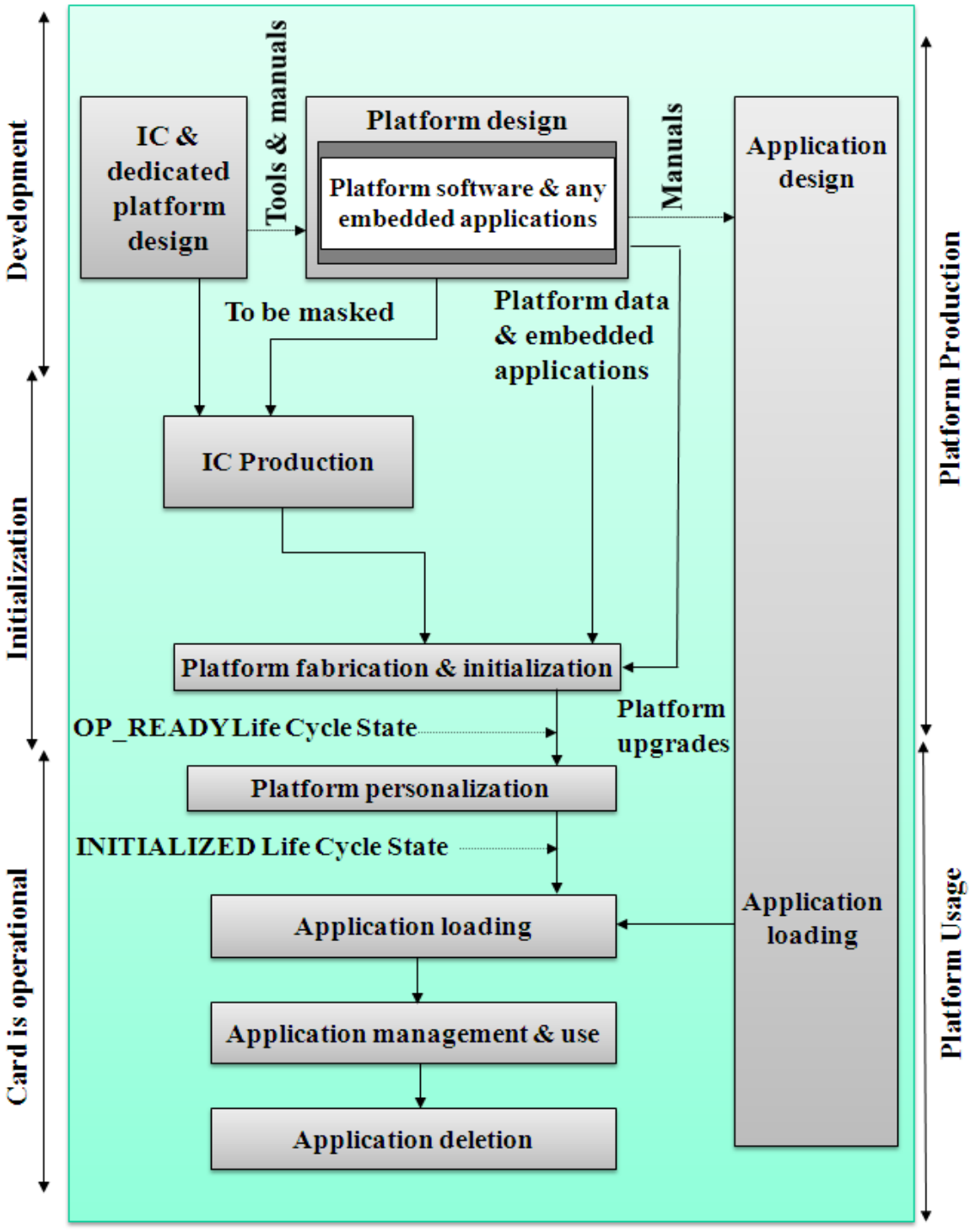

Figure 3: Development and Operation Life Cycle

The scope of this Security Target only includes the Platform Usage phase of the smart card's life cycle after the Platform Personalization step (once the card has reached at least the INITIALIZED state). All the other phases are beyond of the scope of this Security Target.

3.3 TOE Intended Usage

Smart cards are mainly used as data carriers that are secure against forgery and tampering. More recent uses also propose them as personal, highly reliable, small size devices capable

of replacing paper transactions by electronic data processing. Data processing is performed by a piece of software embedded in the smart card chip, usually called an application.

The TOE is intended to transform a smart card into a platform capable of executing applications written in a subset of the Java programming language. The intended use of a Java Card platform is to provide a framework for implementing IC independent applications conceived to safely coexist and interact with other applications into a single smart card.

Applications installed on a Java Card platform can be selected for execution when the card interacts with a card reader. The Card Issuer may also use the card reader to enlarge or to restrict the set of applications that can be executed on the Java Card platform, or to enforce other card management policies, like personalizing the application instances or modifying the life-cycle state of either the whole card or one of its application instances. Only the Card Administrator is supposed to perform card management operations: in particular, neither the Cardholder nor any Application Provider is supposed to download Executable Files to, or remove them from, the card without the explicit authorization and participation of the Card Issuer. The Cardholder is only authorized to make use of the services proposed by the application instances that the Card Issuer has created for him or her.

So far, the most important applications concern:

- Financial applications, like credit/debit ones, stored value purse, or electronic commerce, among others;
- Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines;
- Telephony, through the subscriber identification module (SIM) for digital mobile telephones;
- Electronic signature devices for e-government, like applications based on the standard described in [SSCD];
- Electronic passports and other machine-readable transport documents, like those based on the standard described in [MRTD];
- Personal identification for granting access to secured sites or providing identification credentials to participants of an event;
- Secure information storage, like health records, or health insurance cards;
- Loyalty programs, like the “Frequent Flyer” points awarded by airlines.

Application instances may therefore contain other confidentiality or integrity sensitive data than usual cryptographic keys and PIN codes; for instance: passwords, pass-phrases, or personal information like biometric data, health-care information or the balance of an electronic purse are as confidential as the PIN. Such highly sensitive information may co-exist with other kind of information, like the number of points of a fidelity program.

4 Security Problem Definition

4.1 Assets

This section introduces the assets that the TOE shall protect.

Assets may overlap, in the sense that different assets may refer (partially or wholly) to the same piece of information. For example, "a piece of software" may be either a piece of source code (one asset) or a piece of compiled code (another asset), and may exist in various formats at different stages of its development (digital supports, printed paper). This separation is motivated by the fact that a threat may concern one form at one stage, but be meaningless for another form at another stage.

The assets to be protected by the TOE are listed below. They are grouped into the assets under the control of the Card Manager and the assets under the control of the Runtime Environment. Each group of assets is classified in turn according to whether it is data created by and for the user of the TOE (User data) or data created by and for the TOE (TSF data). The kinds of dangers that weigh on each asset are also specified.

4.1.1 Card Management Assets

This section introduces the assets under the control of the Card Manager.

4.1.1.1 User Data

D.APP-CODE

Application code

The code of the applets and libraries loaded on the card and executed by the Runtime Environment.

The on-card executable code of a single application is called an Executable Module. The actual on-card container of one or more Executable Modules is called an Executable File. Before being installed on the card, an Executable File is called an (Executable) Load File.

In Java Card Terminology, an Executable Module is called an Applet class. The container of one or several Applet classes is called a CAP (Converted APplet) file, or a Package. The word *bytecode* is used to refer to the instructions for the Runtime Environment that are contained in a CAP file.

To be protected from unauthorized modification.

D.APP-INST

Application instance

An instance of an Executable Module of one of the Executable Files loaded on the card.

Each application instance provides a collection of services to the users of the smart card.

This asset shall be protected from unauthorized modification.

D.COMMAND

APDU command

The APDU commands that the Security Domains accepts.

An APDU command addressed to the SD contains a request for a card management service. Valid requests come either from the Cardholder or from the Card Administrator.

This asset shall be protected from unauthorized modification. Some specific card management commands, like those containing keys, shall be also protected from unauthorized disclosure.

4.1.1.2 TSF Data

D.SD-SESSION-KEYS

Session keys

The cryptographic keys that the SD uses for ensuring the integrity and origin of card management requests.

This asset shall be protected from unauthorized disclosure and modification.

D.SD-PERSO

Card personalization data

The Card Issuer's data used to personalize the smart card.

During personalization, the cryptographic keys are stored in the Issuer Security Domain (ISD), the on-card representative of the Card Issuer). These keys needed to support several card management functions, like setting up a secure channel with the terminal.

If the card is issued with Supplementary Security Domains (SSD), cryptographic keys of these SD are also personalized.

These assets shall be protected from disclosure and unauthorized modification.

D.GP-REGISTRY

GlobalPlatform's Registry

The information that the OPEN keeps about the Executable Load Files and the installed application instances.

The GP Registry contains the following information about each application instance:

- identifier,
- privileges,
- current life cycle state,
- memory resource quotas,

The current life cycle state of the whole smart card is also part of the GP registry.

This asset shall be protected from unauthorized modification.

D.CVM

Cardholder Verification Method

The data associated with the Cardholder Verification Method (CVM). The CVM is a single global PIN used to authenticate the Cardholder, which can be shared by all the application instances of the card.

The CVM data includes the following items:

- the PIN code used to authenticate the Cardholder,
- the current number of failed authentication attempts,
- the maximum number of authentication attempts.

These assets shall be protected from unauthorized modification. The PIN code of the CVM shall be also protected from unauthorized disclosure.

D.JCS_CODE

The code of the Java Card System.

To be protected from unauthorized disclosure and modification.

4.1.2 Runtime Environment Assets

This section introduces the assets under the control of the Runtime Environment. These assets are those introduced in [JCSPP], with the exception of the D.APP-CODE, which was already introduced in the previous section.

4.1.2.1 User Data

D.APP_C_DATA

Confidential application data

Confidential sensitive data of the applications, like the data contained in an object, a static field of a package, a local variable of the currently executed method, or a position of the operand stack.

This asset shall be protected from unauthorized disclosure.

D.APP_I_DATA

Integrity-protected application data

Integrity sensitive data of the applications, like the data contained in a Java Card object, a static field of a package, a local variable of the currently executed method, or a position of the operand stack of the JCVM.

This asset shall be protected from unauthorized modification.

D.PIN

Application PIN code

Any end-user's PIN that an applet uses to authenticate the Cardholder and its security attributes (try counter and limit). This value may be different from the global CVM under the control of the Card Manager.

These assets shall be protected from unauthorized modification. The PIN value shall also be protected from unauthorized disclosure.

D.APP_KEYs

Application keys

Cryptographic keys owned by the applets.

This asset shall be protected from unauthorized disclosure and modification.

4.1.2.2 TSF Data

D.SOFTWARE

Embedded Software

The instructions that compose the Embedded Software.

The Embedded Software includes the code masked in ROM and the Patch File loaded in the persistent mutable memory of the card during its initialization phase.

This asset shall be protected from unauthorized disclosure and modification.

Application Note:

This asset covers the D.JCS_CODE asset in [JCSPP]. The term *the code of the Java Card System* used in that protection profile has been replaced in this Security Target by the broader term *Embedded Software*, which also includes the code of the Operating System and the code of the Card Manager.

D.JCS_DATA

Runtime data areas

The internal runtime data areas necessary for the execution of the Java Card VM, such as, for instance, the frame stack, the program counter, the class of an object, the length allocated for an array, and any pointer used to chain data-structures.

To be protected from monopolization and unauthorized modification.The internal runtime data areas necessary for the execution of the Java Card VM, such as, for instance, the frame stack, the program counter, the class of an object, the length allocated for an array, any pointer used to chain data-structures.

To be protected from unauthorized disclosure or modification.

D.SEC_DATA

Sensitive data of the Runtime Environment

The runtime security data of the Java Card RE, like, for instance, the AIDs used to identify the installed applets, the currently selected applet, the current context of execution and the owner of each object.

This asset shall be protected from unauthorized modification.The runtime security data of the Java Card RE, like, for instance, the AIDs used to identify the installed applets, the currently selected applet, the current context of execution and the owner of each object.

To be protected from unauthorized disclosure and modification.

D.API_DATA

APIs private data

Private data of the API, like the contents of the private fields of the Java Card classes of the JC and the GP APIs.

This asset shall be protected from unauthorized disclosure and modification.Private data of the API, like the contents of its private fields.

To be protected from unauthorized disclosure and modification.

D.CRYPTO

Cryptographic data

Cryptographic data used in runtime cryptographic computations, like a seed used to generate a key, or the result of a hashing function.

This asset shall be protected from unauthorized disclosure and modification.Cryptographic data used in runtime cryptographic computations, like a seed used to generate a key.

To be protected from unauthorized disclosure and modification.

4.2 Users / Subjects

4.2.1 IT Entities

The IT entities are programs that invoke the services offered by the platform. This includes programs running on the CAD's side on behalf of the Card Administrator as well as applets running on top of the Java Card platform on behalf of the corresponding Application Provider.

4.2.1.1 Subjects associated to the Card Manager

The following subjects are involved in the security policies enforced by the Card Manager.

S.APP

An *application instance* is an on-card entity implementing one of the services offered by the smart card, possibly using the services that GlobalPlatform offers through its API.

The actual set of applications embedded on the card depends on each card and can be dynamically enlarged.

This subject is called S.PACKAGE in [JCSPP]. This name comes from the fact that the Java Card Firewall security policy grants the same privileges to all the applet instances declared in the same package.

S.NAT

A *native application* is an application written in the native language of the platform. A native application is not executed through the Runtime Environment.

S.OPEN

The *OPEN* is the embedded software component in charge of dispatching the APDU commands to the Application instances. This subject acts on behalf of the Card Issuer.

S.SPY

The *Spy* is any subject acting on behalf of the attacker with the purpose of disclosing, replaying, deleting, modifying or permuting the APDU commands sent to card.

S.SD

The *Security Domain* are privileged applications that hold cryptographic keys which can be used to support Secure Channel Protocol operations and/or to authorize card content management functions.

4.2.1.2 Subjects associated to the Runtime Environment

The following subjects introduced in [JCSPP] are involved in the security policies of the Runtime Environment.

S.JCRE

The *JCRE* stands for those software libraries of the TOE written in Java Card and which are executed on behalf of the Card Issuer. This subject is involved in the Java Card Firewall policy of the Runtime Environment.

S.BCV

The *Bytecode Verifier* is a program that verifies the code of the Executable Load Files downloaded on the card. It acts on behalf of the Controlling Authority. This subject is involved in the Card Content Management security policy.

4.3 Threats

This section describes the threats that concerned the TOE. Only the threat concerning jTOP platform are described, but threats from [ICST] must be considered:

- T.PHYS_MANIPULATION
- T.PHYS_PROBING
- T.MALFUNCTION
- T.Leak_Forced
- T.Leak_Inherent
- T.RND
- T.Abuse_Func
- T.MEM_ACCESS

4.3.1 Card Management

This section introduces the threats concerning card management. The general description of the threat is sometimes followed by examples illustrating particular scenarios for it.

Those threats in [JCSPP] that directly concern card management operations are also included in this section, see T.INSTALL and T.DELETION.

T.IMPERSONATE

The attacker may try to impersonate the Cardholder in order to gain access to the services that the card offers to him.

Impersonating the Cardholder amounts to disclosing or guessing the PIN code stored in the CVM.

T.INVALID-INPUT

The attacker may determine security relevant information, cause the card to malfunction or otherwise compromise security through introduction of invalid input.

Invalid input may take the form of operations that are not formatted correctly, requests for information beyond register limits, or attempts to search for possible undocumented commands. Such inputs could be generated at any time during the normal usage of the card, including prior to access authorization, through normal operations. The attack could also make use of invalid data and inappropriate operations, such as commands/functions with requests/formats that are out of range or otherwise non-conforming to the *accepted* usage. The result of such attacks may be a compromise in the security functions, generation of exploitable errors in operation, or release of protected data.

Application Note:

An invalid parameter is a piece of data that is not encoded in the expected format, or which has been forged by the attacker by other means that those defined in the

functional specification of the TOE. An invalid parameter represents a value that should never be used according to the implementation chosen for a particular type of data.

The following are examples of possible invalid parameters concerning security sensitive information:

- a command with one of the INS bytes defined in [GPCS] which does not have the expected values in the P1 and P2 parameters, or the expected TLV structure in its data filed;
- a key of an invalid length;
- a reference to a component of the key which does not exists for the given key;
- a cryptographic algorithm that is not supported by the platform;
- a message to be encrypted or signed that is not correctly aligned with respect to the cryptographic algorithm to be used;
- a PIN code in hexadecimal format that is interpreted as if it was in decimal form;
- a PIN code of an invalid length;
- a byte-encoded record structure which does not respect the expected encoding format, like a byte where some bits are expected to be set with a specific value.

T.INVALID-ORDER

The attacker may corrupt the internal data structures of the platform through the invocation of the functions provided by the interface in an unexpected order.

This threat includes the possibility for a friendly user to invoke the functions implementing a TSF in an order that compromises the security measures they enforce. For instance, the programmer of an application could try to encrypt a piece of information before specifying the cryptographic algorithm to be used. It also covers the possibility that the attacker calls a TSF before this function has been correctly created and has initialized its internal data structures, exploiting the security breaches exposed by an incoherent or outdated state of the TSF.

Application Note:

The following list illustrates some possible scenarios for this kind of attack:

- The attacker may send to the card a suite of commands that is not in the expected order, like for example:
  - a LOAD command that is not preceded by an INSTALL one,
  - a sequence of STORE DATA commands personalizing the SD data where the commands are not arranged in the expected order,
  - a SELECT command sent to an application that has not been installed yet.
- The attacker may require performing an authentication operation before all the parameters required by the authentication service have been supplied. For instance, the attacker may try to be authenticated as the Cardholder before the CVM service has been initialized with a PIN code.
- The attacker may try to consult the value of a key after the end of its lifetime.

T.FORCED-RESET

The attacker may force the card into an insecure life cycle state through inappropriate termination of selected operations.

Attempts to generate a non-secure life cycle state in the card may be made through premature termination of transactions or communications between the card and the card

reading device, by insertion of interrupts, or by selecting deleted applications that were not able to remove its objects. An attacker may also corrupt security sensitive data by provoking the interruption of the execution of the TSF. This includes tearing the smart card from the CAD as well as firing any other mechanism that could stop or deviate the normal execution of a TSF, like hardware interruptions, input/output interruptions, etc.

Application Note:

The following list illustrates some possible scenarios for this kind of attack:

- The attacker tires to interrupt the execution of an unsuccessful PIN-code verification before the number of remaining attempts has been updated, so that this value is actually never decremented.
- The attacker tries to interrupt the loading of one of the secret keys of the card.
- The attacker tries to interrupt the copy of a secret key into the buffer used by hardware cryptographic functions, in order to shrink the length of the key during an encryption operation.

T.REPLAY

The attacker may penetrate on-card security through the reuse of a completed (or partially completed) operation that was previously performed by an authorized user.

A completed (or partially completed) operation may be replayed in an attempt to bypass security mechanisms or to expose security-related information. For instance, the attacker may try to send to the card an APDU command that he intercepted in a previous session.

The attacker may also use authentication information that was previously delivered to him in order to disclose or modify a piece of information stored in the card that is currently in use by other applications. For instance, the attacker may use authentication information that was once valid, but that is not longer valid, like an old PIN value or cryptographic key.

Application Note:

The following list illustrates some possible scenarios for this kind of attack:

- The aim of the attacker may be, for example, to perform the deletion of an application instance, or the loading of malicious applications. An example of a sequence of commands that could be intercepted for replay is the original INSTALL[for load], LOAD and INSTALL[for install] commands used to create an application instance that has been deleted. If the application had been deleted because it contains security vulnerabilities, the reloading might enable the attacker to continue to exploit that vulnerability.
- The attacker tries to use a previous session key in order to decrypt or falsify a message sent to the card.
- The attacker tries to intercept a command containing a secret key to be loaded on the card, guess the key by some means, and replay the intercepted message later on in order to set the broken key again.
- The attacker tries to re-play an outdated PIN code that was formerly used by the Cardholder.

T.BRUTE-FORCE

The attacker may search the entire user-accessible data space to identify platform and application data.

APDU commands (API methods) can be repetedly transmitted (invoked) to attempt the brute force extraction of secrets such as cryptographic keys or PINs. This threat is distinguished by the use of valid commands with valid range requests that are repeated to reveal as much as possible of the data space. For example, an attacker may systematically experiment with different forms of input. The attack is based on the black box software engineering technique of establishing the nature of algorithms and predicates. If carried out exhaustively it could facilitate the reverse engineering of particular applications as well as the extraction of operational and security related information. The attack could also generate errors in the operation of the card. It may make use of the same valid command with valid range requests repeated many times.

Application Note:

The following list illustrates some possible scenarios for this kind of attack:

- The attacker may search the entire space of keys, signatures or message authentication codes to decrypt, falsify or silently modify a piece of data.
- The attacker may search the entire space of PIN codes in order to detect the one identifying the Cardholder.

T.LIFE-CYCLE

The attacker may exploit commands that were necessary for another part of the card life cycle but are not currently allowed, to expose TSF data or sensitive application data.

Certain card management commands may not be required or allowed in the specific phase of operation being executed. Examples include use of commands which have no operational use, but which could be used to test or debug the internal interfaces of the systems implementing the TOE.

Application Note:

The following list illustrates some possible scenarios for this kind of attack:

- The attacker tries to load an Executable File before the keys required for opening a Secure Channel with the terminal have been populated.
- The attacker tries to apply a patch to the code of the TOE after card initialization;
- The attacker tries to modify the card contents when the card has been temporarily locked;
- The attacker tires to execute an application instance that has been temporarily disabled or definitely deleted.

This attack covers the T.Abuse-Func introduced in [ICPP]: *An attacker may use functions of the TOE which may not be used after TOE Delivery in order to (i) disclose or manipulate User Data, (ii) to manipulate (explore, bypass, deactivate or change) security features or functions of the TOE or of the Smartcard Embedded Software or (iii) to enable an attack.*

T.RECEIPT

If a receipt is requested by the Card Administrator, the attacker may generate fake receipts in order to hide or falsify completion proofs of card content management operations.

This threat includes the possibility of software attacks on the card or cryptographic attacks.

4.3.2 Runtime Environment

This section introduces the threats to the assets under the control of the Runtime Environment. Several groups of threats are distinguished according to the means used in the attack. The classification is also inspired by the components of the TOE that are supposed to counter each threat.

All the threats introduced in this section come from [JCSPP].

4.3.2.1 Cryptography

T.CRYPTO

The attacker may defeat the TSFs through a cryptographic attack against the algorithm or through a brute-force attack on the function inputs.

This attack involves any cryptographic function including encode/decode functions, signature functions or random number generators. The attacker's goal is either to exploit a weakness in the algorithm itself or in the way it was implemented, or, through brute-force substitutions, to find the appropriate keys and inputs. Weaknesses in cryptographic algorithms include both vulnerabilities raising from the theoretical ground on which the algorithm relies and weaknesses possibly involving information leakage during the operation with cryptographic keys that could be observed through SPA, DPA, DFA, or EMA techniques. The attacker's ultimate goal is to disclose sensitive platform or application data.

4.3.3 Java Card System Protection Profile - Open Configuration

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required. Several groups of threats are distinguished according to the configuration chosen for the TOE and the means used in the attack. The classification is also inspired by the components of the TOE that are supposed to counter each threat.

4.3.3.1 CONFIDENTIALITY

T.CONFID-JCS-CODE

The attacker executes an application without authorization to disclose the Java Card System Code.

This threat concerns logical attacks at runtime in order to gain a read access to executable code, typically by executing an application that tries to read the memory area where a piece of platform code is stored.

Directly threatened asset(s): D.JCS_CODE.

Application Note:

The term *Java Card System* used in [JCSPP] has been replaced in this Security Target by the broader term *Embedded Software*, which also includes the Card Manager and the Operating System.

T.CONFID-APPLI-DATA

The attacker executes an application to disclose data belonging to another application without its authorization.

This concerns logical attacks at runtime in order to gain read access to the class instances and the arrays created by the other application instances.

Directly threatened asset(s): D.APP_C_DATA, D.PIN and D.APP_KEYs.

T.CONFID-JCS-DATA

The attacker executes an application to disclose (private) data belonging to the Java Card System.

This concerns logical attacks at runtime in order to gain read access to the private data of the Runtime Environment, the Operating System or the Card Manager. Private data of the Runtime Environment includes TSF data contained in the runtime data areas of the Java Card Virtual Machine and the private fields of the classes implementing the Java Card API.

Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA and D.CRYPTO.

Application Note:

The term *Java Card System* used in [JCSPP] has been replaced in this Security Target by the broader term *Embedded Software*, which also includes the Card Manager and the Operating System.

4.3.3.2 INTEGRITY

T.INTEG-APPLI-CODE

The attacker executes an application to alter (part of) its own code or another application's code.

This concerns logical attacks at runtime in order to gain write access to the memory zone where executable code is stored.

Directly threatened asset(s): D.APP_CODE.

T.INTEG-APPLI-CODE.LOAD

The attacker modifies (part of) its own or another application code when an application package is transmitted to the card for installation.

This threat concerns the modification of application code in transit to the card.

Directly threatened asset(s): D.APP_CODE.

T.INTEG-JCS-CODE

The attacker executes an application to alter (part of) the Embedded Software.

This concerns logical attacks at runtime in order to gain write access to executable code.

Directly threatened asset(s): D.JCS_CODE.

Application Note:

The term *Java Card System* used in [JCSPP] has been replaced in this Security Target by the broader term *Embedded Software*, which also includes the Card Manager and the Operating System.

T.INTEG-JCS-DATA

The attacker executes an application to alter (part of) Java Card System or API data.

This concerns logical attacks at runtime in order to gain write access to the private data managed by the Java Card Runtime Environment, the Java Card Virtual Machine and the private data of Java Card API classes.

Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA and D.CRYPTO.

Application Note:

The term *Java Card System* used in [JCSPP] has been replaced in this Security Target by the broader term *Embedded Software*, which also includes the Card Manager and the Operating System.

T.INTEG-APPLI-DATA

The attacker executes an application to alter (part of) another application's data.

This threat concerns logical attacks at runtime in order to gain unauthorized write access to application data.

Directly threatened asset(s): D.APP_I_DATA, D.PIN and D.APP_KEYs.

T.INTEG-APPLI-DATA.LOAD

The attacker modifies (part of) the initialization data contained in an application package when the package is transmitted to the card for installation.

This threat concerns the modification of the values to be used for initializing the static fields defined in the Executable Load File. It also covers the personalization data that the Card Administrator sends to an installed applet during applet personalization.

Directly threatened asset(s): D.APP_I_DATA and D_APP_KEY.

Other attacks are in general related to one of the above, and aimed at disclosing or modifying on-card information. Nevertheless, they vary greatly on the employed means and threatened assets, and are thus covered by quite different objectives in the sequel. That is why a more detailed list is given hereafter.

4.3.3.3 IDENTITY URSUPATION

T.SID.1

The attacker either impersonates an on-card subject or an external user of the card.

In order to impersonate an on-card subject, the attacker may execute an applet instance which impersonates another applet instance, or even the JCRE. In this case, the attacker's aim is to gain illegal access to the internal resources that the Runtime Environment provides. The attacker may also try to impersonate an external user of the card, like a Card Administrator.

This attack concerns the fraudulent adoption of special privileges only granted to the Runtime Environment or to one of the applet instances registered on the card, and specially the default and the currently selected ones. It also concerns the possibility of confusing an external user of the smart card, who may think that he is communicating with one of his on-card representatives, while he is actually communicating with the attacker.

Directly threatened asset(s): D.SEC_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the JCRE is usurped), D.PIN and D.APP_KEYs.

Application Note:

This threat refines the corresponding one in [JCSPP] by specifying the *resources* that can be accessed and the *end users* that the attacker could try to impersonate.

T.SID.2

The attacker modifies the identity of the privileged roles.

This threat concerns the fraudulent adoption of a privileged role, like the Default or Selected Applet. The actors embodying these roles have access to platform resources or services that may not be available to other applet instances.

Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged).

Application Note:

The following list illustrates some possible scenarios for this threat:

- The attacker tires to change the life cycle state of another applet or the whole card, to modify the card contents or to perform any other operation which is only allowed to the Card Administrator.
- The attacker executes an application that fraudulently tries to lock or terminate the card without authorization.
- The attacker executes an application that tries to reset the PIN code of the CVM.
- The attacker tries to set a malicious application instance as the default selected one, so that it catches commands that are intended for other application instances.

This threat details the corresponding one in [JCSPP] by stating that the privileged roles that the attacker could try to modify are the ones defined for an applet in [GPCS].

4.3.3.4 UNAUTHORIZED EXECUTION

T.EXE-CODE.1

The attacker executes an applet instance that performs an unauthorized execution of a method.

This threat concerns:

- invoking a method outside the scope of the visibility rules provided by the public/private access modifiers of the Java programming language;
- invoking a method on class instance that its owner did not declare as being shareable with the invoker of the method

Directly threatened asset(s): D.APP_CODE.

T.EXE-CODE.2

The attacker executes an applet instance that performs an unauthorized execution of a method fragment or arbitrary data.

This attack concerns jumping inside a method fragment, or interpreting the contents of a data memory area as if it was executable code.

Directly threatened asset(s): D.APP_CODE.

T.NATIVE

The attacker executes a native method to bypass a security function such as the Java Card Firewall.

The execution of native code is not under the control of the Java Card Virtual Machine, so it must be secured to prevent bypassing the TSFs. No distrusted native code may reside on the card.

Directly threatened asset(s): D.JCS_DATA.

4.3.3.5 DENIAL OF SERVICE

T.RESOURCES

An attacker prevents correct operation of the Java Card System through consumption of some resources of the card: RAM or NVRAM.

This attack concerns loading and executing an applet instance that could monopolize all the available smart card memory, so that access to other card services is systematically denied.

Directly threatened asset(s): D.JCS_DATA.

4.3.3.6 APPLET MANAGEMENT

T.INSTALL

The attacker may fraudulently install an Executable File or application instance on the card.

This threat concerns either the installation of bad-formed or ill-typed code, or an attempt to induce a malfunction in the TOE through the installation process, for instance by corrupting the Executable File during its installation. It also concerns the attribution of abusive privileges during the activation of an Executable Module that has been licitly loaded on the card.

If the card has been set up to the closed state, this threat also concerns any attempt from the attacker to load additional Executable Files in the card or creating new applet instances.

Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application).

T.DELETION

The attacker deletes an installed Executable File or application instance that is already in use by some other file or application instance on the card, or uses the delete functions to pave the way for further attacks by putting the TOE in an insecure state.

Insecure states could be the result of broken references to garbage collected code or data, leading to information containers that have been reused by the platform for other purposes.

Directly threatened asset(s): D.SEC_DATA and D.APP_CODE.

Application Note:

The following list illustrates some possible scenarios for this kind of attack:

- The attacker tries to delete an application instance that has shared some of its class instances with another application instances;
- The attacker tries to delete an Executable File which contains a library of the JCAPI or the GPAPI that is necessary for the correct execution of the TOE;
- The attacker tries to delete a SD;

- The attacker tries to delete an Executable File in order to free its allocated memory blocks, and then try to gain access to the information contained in that block through the allocation functions.

4.3.3.7 SERVICES

T.OBJ-DELETION

The attacker keeps a reference to a garbage collected object in order to force the TOE to execute an unavailable method, to make it crash, or to gain access to a memory containing data that is now being used by another application.

The aim of the attacker is to get access to services that the service provider believes no longer available, or to get unauthorized access to private information owned by other applets. For instance, the attacker could try to read the contents of a newly allocated array with the hope that it still contains residual information that was previously stored in a garbage-collected object. If the garbage collector introduces hanged references, the attacker could exploit those references to access a memory block being used by another application.

Directly threatened asset(s): D.APP_C_DATA, D.APP_I_DATA and D.APP_KEYs.

4.3.3.8 MISCELLANEOUS

T.PHYSICAL

The attacker discloses or modifies the design of the TOE, its sensitive data or application code by physical (opposed to logical) tampering means. This threat includes IC failure analysis, electrical probing, unexpected tearing, and DPA. That also includes the modification of the runtime execution of Java Card System or SCP software through alteration of the intended execution order of (set of) instructions through physical tampering techniques.

This threatens all the identified assets.

This threat refers to the point (7) of the security aspect #.SCP, and all aspects related to confidentiality and integrity of code and data.

4.4 Organisational Security Policies

This section describes the security policies. As the TOE is a composite TOE, the [ICST] OSP must be considered:

- P.PROCESS-TOE
- P.Add-Functions

4.4.1 Embedded Software OSP

This Security Target enlarges the organizational security policies introduced in [JCSPP] with some of the policies introduced in GlobalPlatform's specifications.

OSP.FILE-ORIGIN

Only Card Administrators are allowed to transmit new Executable Load Files to the card.

If the card has not been configured to perform DAP verification, then Card Administrators shall only download Executable Load Files received from the Controlling Authority through

the secure communication channel linking them. This rule only applies if the Controlling Authority and the Card Administrator are separate roles.

OSP.SECRETS

Only the Platform Manufacturer and Card Administrators may load secrets protecting the assets on the smart card, such as cryptographic keys and PIN codes. Such secrets shall be generated, distributed and stored off-card, destroyed and exported to the card in a secure manner, which prevents the attacker to obtain them from the IT or non-IT environment. PIN codes shall be transmitted to the Cardholder using a secure channel that ensures its origin, integrity and confidentiality. This rule also applies to keys that are not imported but automatically generated on-card by the applets. In this case it is up to the Controlling Authority to check that such keys are securely generated, distributed, stored and destroyed.

The Controlling Authority shall generate, store, and destroy the private key that it uses for DAP signature in a secure manner.

Application Note:

PIN codes concern both the global PIN that the Card Manager implements as well as any specific PIN created by the installed applets.

OSP.KEY-LENGTH

The Controlling Authority shall inspect the code of the applets and only validate those respecting the constraints regarding cryptographic key lengths provided in the Security Functional Requirements of this document.

Should an applet use the simple DES algorithm that the platform offers through the Java Card API, the Controlling Authority shall ensure that the applet's algorithm chains several simple DES operations, so that to provide DES keys of at least 112 bits.

Application Note:

The precise list of cryptographic algorithms defined in the Java Card API that fall into the scope of evaluation is summarized in the document [PROFILE].

OSP.NO-RMI-APPLETS

The Controlling Authority shall not validate applets relying on the Remote Method Invocation (RMI) mechanism.

The Controlling Authority is supposed to carefully inspect its bytecode, possibly with the help of static analysis tools, and to reject it if it contains a piece of code that relies on the following classes and interfaces of the Java Card API: `RMI`, `RMIService` and `CardRemoteObject`.

OSP.PERSONALIZATION

Until the card is successfully moved to a state where all the security functions are enabled, it shall be under the physical control of the Card Enabler, and shall be only used in a secure environment. Once the ISD is successfully transitioned to such state, the card shall be placed under the administrative control of the Card Issuer.

The card is issued to the Cardholder only after reaching the SECURED life cycle state described in GlobalPlatform's specifications.

4.4.2 Java Card System Protection Profile - Open Configuration

OSP.VERIFICATION

Before loading an Executable Load File on the card, the Controlling Authority shall ensure that it respects the security recommendations stated in [USR]. In particular, the Controlling Authority shall check that the Executable Load File successfully passes bytecode verification using Export Files that match the CAP files that are already installed on the card. Bytecode verification shall include:

- well-formedness of the CAP file structure and verification of the typing constraints on its bytecodes,
- binary compatibility with installed Executable Files and the assurance that the export files used to check the Executable Load File correspond to those that will be present on the card when loading occurs.

Upon successful verification of an Executable Load File, all the roles involved in card content management shall immediately activate all the IT and organizational measures required for preventing any modification of it until it is downloaded into the card. If the Card Manufacturer has configured the card to verify DAP signatures, then the Controlling Authority shall electronically sign the file immediately after successful verification.

If the card has not been configured to verify DAP signatures, the Controlling Authority shall transmit the Executable Load File to the Card Administrator through a secure communication channel ensuring the origin and the integrity of transmitted files. Upon reception the Card Administrator shall store the Executable File in its secure environment until the file is downloaded into the card. Obviously, this rule applies only when the Controlling Authority and the Card Administrator are separate roles. This policy shall ensure the consistency between the export files used in the verification and those used for installing the verified file. The policy must also ensure that no modification of the file is performed in between its verification and the signing by the Controlling Authority. See #.VERIFICATION for details. If the application development guidance provided by the platform developer contains recommendations related to the isolation property of the platform, this policy shall also ensure that the Controlling Authority checks that these recommendations are applied in the application code.

4.5 Assumptions

This section describes the assumptions for the TOE. the assumptions from [ICST] must be considered:

- A.Process-Sec-IC
- A.Resp_App
- A.Plat_Appl
- A.Key_Functions

4.5.1 Assumptions on the Embedded Software

The scope of the Embedded Software addressed in this Security Target has been enlarged with respect to [JCSPP] so as to include the Card Manager. As a consequence, the assumption A.CARD-MANAGEMENT included in that protection profile is no longer pertinent, and it is discharged by the threats considered in the following section. The A.NATIVE assumption has been also adapted to the TOE defined in this Security Target, which includes the whole Java Card API, including native methods.

Some extra assumptions about the behavior of the actors involved in the card manufacturing and card administration processes are also included to ensure that the security objectives defined in this Security Target are sufficient for countering those broader threats.

A.NATIVE

All pre-issuance native application on the card are assumed to be compliant with the TOE so as to ensure that security policies and objectives described herein are not violated.

Application Note:

In [JCSPP], this assumption also supposes that native methods of the Java Card API also enforce the security policies defined for the platform. This part of the assumption has been discharged in this Security Target, as the native libraries of the Operating System that support API implementation are also in the scope of this TOE.

4.5.2 Java Card System Protection Profile - Open Configuration

This section introduces the assumptions made on the environment of the TOE.

A.APPLET

The Executable Files loaded in the post-issuance phase do not contain native methods. The Java Card specification explicitly does not provide any support for native methods apart from those of the Java Card API.

A.VERIFICATION

All the bytecodes of the Executable Files masked on the card have successfully passed the Bytecode Verification process and have not been modified after being verified. Moreover, they only contain applets that follow the security recommendations stated in [USR].

5 Security Objectives

5.1 Security Objectives for the TOE

This section describes the security objectives for the TOE. The objectives described are those from the jTOP platform, but the objectives for the TOE from [ICST] must be considered. Here is the list of the security objectives from [ICST]:

- O.Add_Function
- O.Mem_Access
- O.Leak_Inherent
- O.Leak_Forced
- O.Phys_Probing
- O.Phys_Manipulation
- O.Malfunction
- O.Identification
- O.Abuse_Func
- O.RND

5.1.1 Objectives for the Card Manager

This section introduces the security objectives that are relative to card management.

5.1.1.1 Communication with the terminal

O.REQUEST

The TOE shall reject any card management request containing data that is not in the expected format.

In particular, those APDU commands which are ill-formed with respect to the functional specification of the TOE shall not be processed. Values of type short and byte which are outside of the expected range shall be also rejected by the methods of the JC and GP APIs.

O.INFO-ORIGIN

The TOE shall authenticate the origin of the card management requests that the card receives, and authenticate itself to a Card Administrator.

The goal is to ensure that the information modifying the security attributes of the card comes from a trustable actor who has carefully analyzed the consequences of the card management operation, namely, a Card Administrator. In the other way round, the on-card representative of a Card Administrator shall authenticate itself to that privileged user in order to prevent releasing sensitive information to a malicious applet.

Application Note:

This objective generalizes the O.LOAD and O.INSTALL objectives in [JCSPP]. The verification of the origin applies to all card management operations, and not only to the loading of new Executable Files.

O.INFO-CONFIDENTIALITY

The TOE shall be able to process confidential requests containing encrypted data.

The goal is to prevent the disclosure of the secret keys enabling to open a Secure Channel between the SD and the Card Administrator. In addition to this, the Application Providers may want to protect the code of their applications while they are in transit to the card. Finally, application instances may also request the SD to provide a Secure Channel enforcing confidentiality for their own personalization.

O.NO-KEY-REUSE

The TOE shall ensure that session keys can be used only once.

The keys used to ensure the origin, integrity and confidentiality of the card management requests shall contain an unpredictable piece of data that is randomly chosen by the TOE, so that they can be valid only within the session in which they are generated.

5.1.1.2 Card Content Management

O.INFO-INTEGRITY

In the operational phase of the card, the TOE shall verify the integrity of the card management requests that the card receives.

The goal is to ensure that the card management operation processed by the card is exactly the one that the Card Administrator requested. In addition to this, each applications installed on the card may also request the SD to verify the integrity of the commands used to personalize it. This only applies to the operational phase of the card, where the card is in a potentially hostile environment.

Application Note:

This objective generalizes the O.LOAD and O.INSTALL objectives in [JCSPP] in the same way as O.INFO-INTEGRITY.

O.CARD-MANAGEMENT

The TOE shall control the access to card management functions such as loading, installation, extradition, deletion of applications, GlobalPlatform registry updates and SD key personalization or replacement.

The card manager, the application with specific rights responsible for the administration of the smart card, shall control the access to card management functions. It shall also implement the card issuer's policy on the card.

The card manager shall prevent that card content management is carried out, for instance, at invalid states of the card or by non-authorized actors.

Application Note:

This security objective for the TOE corresponds to security objective for the environment OE.CARD-MANAGEMENT from the Java Card Protection Profile.

The card manager will be tightly connected in practice with the rest of the TOE, which in return shall very likely rely on the card manager for the effective enforcement of some of its security functions.

The mechanism used to ensure authentication of the TOE issuer, that manages the TOE, or of the Service Providers owning a Security Domain with card management privileges is a secure channel. This channel will be used afterwards to protect commands exchanged with the TOE in confidentiality and integrity.

The platform guarantees that only the ISD or the Service Providers owning a Security Domain with the appropriate privilege (e.g Delegated Management) can manage the applications on the card associated with its Security Domain. This is done accordingly with the card issuer's policy on card management.

The actor performing the operation must beforehand authenticate with the Security Domain. In the case of Delegated Management, the card management command will be associated with an electronic signature (GlobalPlatform token) verified by the ISD before execution.

O.RECEIPT

The TOE shall generate on request non-repudiable receipts of the completion of card content management operations.

5.1.1.3 Life Cycle Management

O.LIFE-CYCLE

The TOE shall enforce a well-defined life-cycle, keeping track of its current state, and controlling that the operations required by the users are consistent with the current life cycle state of the TOE.

The life cycle shall include a well-identified life cycle state after which the TOE is in a secure state and immediately operational, and prevent the execution of those functions that could become insecure. Examples of functions that would downgrade the security of the TOE in the operational state are testing and debugging functions or loading a Patch File after platform initialization. The life cycle shall also include a final state, where the TOE is definitely terminated.

5.1.1.4 Cardholder Verification Method

O.GLOBAL-CVM

The TOE shall enable the applications to consistently manage a unique service for authenticating the Cardholder (CVM service).

The TOE shall restrict the modification of the security attributes of the CVM only to some privileged applications appointed by the Card Administrator. Only the Card Administrator may grant the CVM privilege to an applet.

O.CVM-BLOCK

No further Cardholder authentication attempts shall be possible once the maximal number of attempts has been reached, until a special action is performed by a privileged user.

5.1.1.5 Logical Protection

O.ERROR-COUNTERS

The TOE must prevent the release of information through the analysis of responses to repetitive stimulations. This objective could also work through the detection of such attacks and the initiation of corrective actions to counter such attempts.

Detection involves journalizing how many times the same unsuccessful operation has been carried out, like the number of unsuccessful authentication attempts made by the Cardholder. Corrective actions involve temporarily or definitely locking some of the services that the card provides.

O.RECOVERY

The TOE shall check at the beginning of each session whether there still remained some unfinished tasks when the card was powered off. If there are unfinished tasks, the TOE shall either complete them (if possible), or abort the whole action that gave rise to them and roll back to a safe state.

If power is lost or if the smart card is withdrawn from the CAD while an atomic operation is in progress, the TOE must eventually complete the interrupted operation successfully, or recover to a consistent and secure state. Atomic operations are, for instance, loading new secret keys, downloading an Executable File, or installing or deleting an applet instance.

Application Note:

In the [JCSPP], this objective is part of the objectives for the security environment. In this Security Target it has been included among the security objectives for the TOE because the scope of the TOE includes the Operating System, which implements atomic transactions.

O.LOCK

The TOE shall enable the applications to temporarily or definitely disable the services they provide, or even the services provided by the whole platform. The TOE shall restrict the use of such sensitive capabilities only to privileged applications appointed by the Card Administrator.

Entering into a state where the set of services is restricted is a countermeasure that the applets may use in response to a security violation. This counter-measure has to be restricted to appointed applications in order to prevent a possible denial of service attack.

5.1.2 Objectives for the Runtime Environment

This section introduces the security objectives that are relative to the runtime environment on which the applets are executed.

5.1.2.1 Execution of applets

The security objectives in this section concern the way in which the code of the applet instances is executed.

O.VERIFICATION

The TOE shall enforce runtime verifications to ensure the adequacy of each bytecode operand to the intended semantics of that bytecode.

Runtime verifications shall include checking at least the following properties:

- bytecode instructions represent a legal set of instructions used on the Java Card platform;
- partial adequacy of bytecode operands to bytecode semantics;
- absence of operand stack overflow/underflow;

- monitoring of control flow confinement to the current package code (detection of control jumps outside the current Method Component);
- forged references to a class instance or an arrays are rejected;
- illegal offsets into a class instance, array, static field image or local variable are rejected;
- native method invocation is restricted to the set of reliable ones masked with the Embedded Software.

Application Note:

This objective is close to the O.VERIFICATION objective included in [JCSPPD]. The verifications that jTOP's defensive virtual machine enforce are not as large as the ones provided by an on-card bytecode verifier, but they nevertheless ensure the main security properties expected from it.

O.OS-SUPPORT

The Operating System layer shall include low-level support to the TSF of the Java Card Runtime Environment.

The support that the TOE shall provide to the Java Card Runtime Environment includes:

- Appropriate use of the MMU for protecting TSF data and TSF code against disclosure or modification by the applet instances.
- Atomically updating a collection of persistent memory positions;
- Detecting segmentation faults for the blocks allocated by the JCVM.

5.1.3 Java Card System Protection Profile - Open Configuration

This section defines the security objectives to be achieved by the TOE.

5.1.3.1 IDENTIFICATION

O.SID

The TOE shall uniquely identify every applet instance before granting it access to any security sensitive service.

The TOE shall only accept requests coming from application instances that have been correctly registered by the OPEN in the GlobalPlatform's registry. The security functions offered through the Java Card or GlobalPlatform's API shall always reject requests coming from applications that have not yet been registered.The TOE shall uniquely identify every subject (applet, or package) before granting it access to any service.

Application Note:

An application instance that is not yet registered cannot share objects with other application instances, nor set the PIN code of the CVM. As a consequence, those services are rejected when invoked from the `Applet.install` method before that method registers the applet instance.

5.1.3.2 EXECUTION

O.FIREWALL

The TOE shall provide controlled sharing of data containers owned by Applet instances of different Executable Files, and between applet instances and the TSF.

The Platform shall ensure controlled sharing of class instances and arrays, and isolation of the data and the code of different Executable Files (that is, controlled execution contexts).

An applet instance shall neither read, write nor compare a piece of data belonging to an instance of another applet that is not in the same context, nor execute one of the methods of an instance of an applet in another context without its authorization.

O.GLOBAL_ARRAYS_CONFID

The TOE shall ensure that the APDU buffer that is shared by all applications is always cleaned upon applet selection.

The TOE shall ensure that the global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

O.GLOBAL_ARRAYS_INTEG

The TOE shall ensure that only the currently selected applications may have a write access to the APDU buffer and the global byte array used for the invocation of the install method of the selected applet.

O.NATIVE

The *only* means that the JCVM shall provide for an applet instance to execute native code is the invocation of a method of the Java Card API.

Because the direct execution of arbitrary native code is beyond the control of the JCVM, it must be secured so as not to provide ways to bypass the TSFs. No distrusted native code may reside on the card. Loading of native code into the Platform is submitted to the same requirements.

O.OPERATE

The TOE must ensure continued correct operation of its security functions.

The TOE must also return to a well-defined valid state before a service request in case of failure during its operation.

O.REALLOCATION

The TOE shall ensure that the re-allocation of a memory block for the runtime areas of the JCVM does not disclose any information that was previously stored in that block.

O.RESOURCES

The TOE shall control the availability of resources for the application instances.

The TOE must enforce quotas and limitations in order to prevent unauthorized denial of service or malfunction of the TSF. This concerns both execution (dynamic memory allocation) and installation (static memory allocation) of application instances and Executable Files.

5.1.3.3 SERVICES

O.ALARM

The TOE shall provide appropriate feedback information upon detection of a potential security violation.

This concerns in particular the security exceptions thrown by the JCVM, or any other security-related event occurring during the execution of a TSF.

O.CIPHER

The TOE shall provide a means to cipher sensitive data for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards.

Ciphering includes the following cryptographic operations: data encryption and decryption, electronic signature generation and verification, computation and update of a hash value, computation and update of a checksum and random number generation. These operations shall include mechanisms to resist to the SPA/DPA/DFA/EMA techniques that are part of the state of the art. The cryptographic services shall also ensure that only keys that are consistent with the specified algorithm are used, and prevent any use of the services before it has been correctly initialized.

O.KEY-MNGT

The TOE shall provide means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys, including the following points:

- Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes,
- Keys shall be distributed in accordance with specified cryptographic key distribution methods. In particular, keys must be loaded through a secure channel ensuring key origin, authenticity and confidentiality.
- Keys shall be initialized before being used. In particular, the SD shall verify that all the components of a loaded DES key have been received and checked before using the key.
- Keys shall be stored in secure containers that prevent their disclosure by direct observation of the smart card memory (memrory dumping)
- Keys shall be destroyed in accordance with specified cryptographic key destruction methods. These methods shall include the possibility of limiting the lifetime of the key value to the current session.

O.PIN-MNGT

The TOE shall provide a means to securely manage PIN objects.

Secure management of PIN objects includes:

- Atomic update of PIN code and of the try counter,
- No rollback of the number of unsuccessful authentication attempts,
- Encryption of the PIN value and no clear-PIN-reading function,
- Enhanced protection of the PIN's security attributes,
- Software countermeasures to make it difficult to observe PIN comparisons.

Application Note:

PIN objects may play key roles in the security architecture of client applications. The way they are stored and managed in the memory of the smart card must be carefully considered, and this applies to the whole object rather than the sole value of the PIN. For instance, the try counter's value is as sensitive as that of the PIN.

O.TRANSACTION

The TOE must provide a means to atomically execute a set of operations atomically.

O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION and O.CIPHER are actually provided to applets in the form of Java Card APIs. Vendor-specific libraries can also be present on the card and made available to applets; those may be built on top of the Java Card API or independently. These proprietary libraries will be evaluated together with the TOE.

To atomically execute a sequence of operations means that either all the operations are completely executed or the Runtime Environment behaves as if none of them would be executed. Applet instances may request the platform to perform a sequence of modifications atomically through the Java Card API.

5.1.3.4 OBJECT DELETION

O.OBJ-DELETION

The TOE shall ensure the object deletion shall not break references to objects. It also shall ensure that object deletion shall not introduce dangling pointers, and that garbage-collected information is always cleared.

Object de-allocation should not introduce security holes in the form of references pointing to memory zones that are not longer in use, or have been reused for other purposes. This process should not be maliciously used to circumvent the TSF, like the Firewall. Erasure, if deemed successful, shall ensure that the deleted class instance is no longer accessible and that its contents have been cleared.

5.1.3.5 APPLET MANAGEMENT

O.DELETION

The TOE shall ensure that both applet and package deletion perform as expected. It shall ensure that both application and Executable File deletion are safe.

The deletion mechanism shall consider the following issues:

- Deletion of installed applets (or Executable Files) shall neither introduce security holes in the form of broken references to garbage collected code or data, nor alter the integrity or confidentiality of the remaining applets. The deletion procedure shall not be maliciously used to bypass the TSF.
- Erasure, if deemed successful, shall ensure that any data owned by the deleted applet is no longer accessible (shared objects shall either prevent deletion or be made inaccessible). A deleted applet cannot be selected or receive APDU commands. The deletion of an Executable File shall make its code no longer available for execution.
- Power failure or other failures during the process shall be taken into account in the implementation so as to preserve the TSPs. This does not mandate the whole process to be atomic, but rather that it can be sliced into small and atomic deletion

steps. For instance, an interrupted deletion may result in the loss of user data, as long as it does not violate the TSPs.

- It shall not be possible to delete the Executable Files corresponding to Java Card and GlobalPlatform's API.

Application Note:

Logical deletion is only acceptable for those Executable Files stored in ROM. Deleted Executable Files stored in EEPROM shall be physically removed from the smart card memory.

O.LOAD

The procedure of loading and installing an Executable Load File shall ensure the integrity and authenticity of the file. The TOE shall ensure that the loading of a package into the card is safe. The TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the Controlling Authority. This verification by the TOE shall occur during the loading or later during the install process.

The TOE must check that each Executable Load File actually comes from a Card Administrator, who has previously checked that it is harmless for the other applications installed on the card. Moreover, for further security, the card shall check the DAP signature of the Controlling Authority in charge of performing that checkings. This signature is attached to the Executable Load File.

If the card has been set to the Closed Mode, it must reject any attempt of loading an Executable File, even if this action is required by an external user authenticated as a Card Administrator.

Application Note:

Usurpation of identity resulting from a malicious installation of an applet on the card may also be the result of perturbing the communication channel linking the CAD and the card. Even if the CAD is placed in a secure environment, the attacker may try to capture, duplicate, permute or modify the packages sent to the card. He may also try to send one of its own applications as if it came from the card issuer. Thus, this objective is intended to ensure the integrity and authenticity of loaded CAP files.

O.INSTALL

The TOE shall ensure that the installation of an applet performs as expected. The TOE shall ensure that the installation of an application is safe. The TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the Controlling Authority. If not performed during the loading process, this verification by the TOE shall occur during the install process.

In order to be safe, the installation process must satisfy the following requirements:

- The TOE must be able to undo all the installation steps when the installation fails or is cancelled, whatever the reasons.
- Installing an application must have no effect on the code and data of already installed applets. In particular, the installation of a new application or Executable File shall not hide or make inaccessible any other application or file already existing on the card.

The installation procedure should not be used to bypass the TSFs. It shall be a secure atomic operation, and free of harmful effects on the state of the other applets.

In particular, the set of privileges granted to an application shall be consistent with the intended meaning of each privilege and with the configuration of GP implemented by the platform.

The procedure of installing an application shall ensure the integrity and origin of the installation request, including the privileges granted to the application.

If the card has been set to the Closed Mode, it must reject any attempt of creating a new applet instance, even if this action is required by an external user authenticated as a Card Administrator.

5.2 Security Objectives for the Operational Environment

This section describes the security objectives for the environment. The TOE being a composite TOE (jTOP platform and IC sle78), the objectives for the environment from [ICST] must be considered:

- OE.Plat-Appl
- OE.Resp-Appl
- OE.Process-Sec-IC

5.2.1 Java Card System Protection Profile - Open Configuration

This section introduces the security objectives to be achieved by the environment.

In this security target, the security objective for the environment OE.CARD-MANAGEMENT from the Java Card Protection Profile is moved to a security objective for the TOE (O.CARD-MANAGEMENT).

OE.APPLET

No applet loaded post-issuance shall contain native methods.

OE.SCP.IC

The SCP shall provide all IC security features against physical attacks.

This security objective for the environment refers to the following security aspect:

- It is required that the IC is designed in accordance with a well-defined set of policies and Standards (likely specified in another protection profile), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

OE.SCP.RECOVERY

If there is a loss of power, or if the smart card is withdrawn from the CAD while an operation is in progress, the SCP must allow the TOE to eventually complete the interrupted operation successfully, or recover to a consistent and secure state.

This security objective for the environment refers to the following security aspect: The smart card platform must be secure with respect to the SFRs. Then after a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state.

OE.SCP.SUPPORT

The SCP shall support the TSFs of the TOE.

This security objective for the environment refers to the following security aspects:

(2) It does not allow the TSFs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System.

(3) It provides secure low-level cryptographic processing to the Java Card System.

(4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism.

(5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low-level control accesses (segmentation fault detection).

OE.VERIFICATION

The Card Administrator transmits an Executable Load File to the card only if the application code complies with the security recommendations in [USR], has successfully passed the Bytecode Verification process and has not been modified afterwards.

Bytecode verification shall include:

- well-formedness of the CAP file structure and verification of the typing constraints on its bytecodes,
- binary compatibility with installed Executable Files and the assurance that the export files used to check the Executable Load File match the CAP files that will be present on the card when loading occurs. All the bytecodes shall be verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. See #.VERIFICATION for details. Additionally, the applet shall follow all the recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.

Application Note:

Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

OE.CODE-EVIDENCE

For application code loaded pre-issuance, evaluated technical measures implemented by the TOE or audited organizational measures must ensure that loaded application has not been changed since the code verifications required in OE.VERIFICATION. For application code loaded post-issuance and verified off-card according to the requirements of OE.VERIFICATION, the Controlling Authority shall provide digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification. For application code loaded post-issuance and partially or entirely verified on-card, technical measures must ensure that the verification required in OE.VERIFICATION are performed. On-card bytecode verifier is out of the scope of this TOE.

Application Note:

For application code loaded post-issuance and verified off-card, the integrity and authenticity evidence can be achieved by electronic signature of the application code, after code verification, by the actor who performed verification.

5.2.2 Miscellaneous

OE.NATIVE

The Platform Developper shall ensure that all pre-issuance native applications masked with the code of the platform enforce the security policies and objectives described in this Security Target.

In particular, native applications that handle Java Card objects must respect the Java Card Firewall policy.Those parts of the APIs written in native code as well as any pre-issuance native application on the card shall be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated.

Application Note:

In [JCSPP], this security objective also requires that parts of the API that are implemented as native methods enforce the security policies defined for the platform. That part of the objective has been discharged in this Security Target, as the native libraries of the Operating System that support API implementation are also in the scope of this TOE.

OE.SECRETS

The attacker shall not be able to obtain neither the private key for generating DAP signatures nor any of the PIN codes or secret keys stored in the card from the TOE IT or non-IT environment.

OE.KEY-LENGTH

The Controlling Authority shall only validate those applets that respect the constraints regarding cryptographic key lengths provided in the Security Functional Requirements of this document.

Application Note:

The precise list of cryptographic algorithms defined in the Java Card API that fall into the scope of evaluation is summarized in the document [PROFILE].

OE.NO-RMI-APPLETS

The Card Administrator shall not load applets based on the Remote Method Invocation (RMI) mechanism.

6 Extended Requirements

6.1 Extended Families

6.1.1 Extended Family FCS_RND - Generation of random numbers

6.1.1.1 Description

This family defines quality requirements for the generation of random numbers which are intended to be used for cryptographic purposes.

6.1.1.2 Extended Components

6.1.1.2.1 Extended Component FCS_RND.1

6.1.1.2.1.1 Description

The generation of random numbers requires that random numbers meet a defined quality metric.

There are no management activities foreseen for this component.

There are no actions defined to be auditable for this component.

6.1.1.2.1.2 Definition

FCS_RND.1 Quality metric for random numbers

FCS_RND.1.1 The TSF shall provide a mechanism to generate random numbers that meet [assignment: a defined quality metric].

Dependencies: No dependencies.

6.1.1.2.1.3 Rationale

It was chosen to define FCS_RND.1 explicitly, because Part 2 of the Common Criteria do not contain generic security functional requirements for Random Number generation. Note tha there are security functional requirements in Part 2 of the Common Criteria, which refer to random numbers. However, they define requirements only for the authentication context, which is only one of the possible applications of random numbers.

6.1.1.3 Rationale

This family has been introduced in [SSVG]. An attacker may predict or obtain information about random numbers generated by the TOE for instance because of a lack of entropy of the random numbers provided. Here the attacker is expected to take advantage of statistical properties of the random numbers generated by the TOE without specific knowledge about the TOE's generator. Malfunctions or premature aging are also considered which may assist in getting information about random numbers. To counter this kind of attacks, the TOE must

ensure the cryptographic quality of random number generation. For instance random numbers shall not be predictable and shall have a sufficient entropy. The introduction of this new class enables to specify the quality metric that must be used.

7 Security Requirements

7.1 Security Functional Requirements

This section introduces the security functionalities of the TOE. The TOE is composed of the jTOP platform and a chip from family SLE78. The following list of SFRs are those from the chip security target [ICST]; they must be considered for the composite TOE, but they are not repeated in it. For their detail, please refer to [ICST].

- FRU_FLT.2-IC "Limited fault tolerance"
- FPT_FLS.1-IC "Failure with preservation of secure state"
- FMT_LIM.1-IC "Limited capabilities"
- FMT_LIM.2-IC "Limited availability"
- FAU_SAS.1-IC "Audit storage"
- FPT_PHP.3-IC "Resistance to physical attack"
- FDP_ITT.1-IC "Basic internal transfer protection"
- FPT_ITT.1-IC "Basic internal TSF data transfer protection
- FDP_IFC.1-IC "Subset information flow control"
- FCS_RNG.1-IC "Quality metric for random numbers"
- FPT_TST.2-IC "Subset TOE security testing"
- FDP_ACC.1-IC "Subset access control"
- FDP_ACF.1-IC "Security attribute based access control"
- FMT_MSA.1-IC "Management of security attributes"
- FMT_MSA.3-IC "Static attribute initialisation"
- FMT_SMF.1-IC "Specification of Management functions"
- FCS_COP.1-IC "Cryptographic support" --> FCS_COP.1-IC/DES, FCS_COP.1-IC/AES, FCS_COP.1-IC/RSA, FCS_COP.1-IC/ECDSA, FCS_COP.1-IC/ECDH, FCS_COP.1-IC/SHA.
- FCS_CKM.1-IC "Cryptographic key management"
- FDP_SDI.1-IC "Stored data integrity monitoring
- FDP_SDI.2-IC "Stored data integrity monitoring and action"

The TOE does not provide RMI functionality option, therefore RMI related entities of the PP Java Card [JCSPP] (Subject, Object, Information, Security Attribute and Operation) and their corresponding SFRs are excluded from the ST.

7.1.1 Java Card System Protection Profile - Open Configuration

This section states the security functional requirements for the Java Card System - Open configuration. For readability and for compatibility with the original Java Card System Protection Profile Collection - Open Configuration [JCSPP], requirements are arranged into groups. All the groups defined in the table below apply to this Protection Profile.

Group	Description

Group	Description
Core with Logical Channels (*CoreG_LC*)	**The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 3.0.4 feature. This group is the union of requirements from the Core (*CoreG*) and the Logical channels (*LCG*) groups defined in [PP/0305] (cf. Java Card System Protection Profile Collection [PP JCS]).**
Installation (*InstG*)	**The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution.**
Applet deletion (*ADELG*)	**The ADELG contains the security requirements for erasing installed applets from the card, a feature introduced in Java Card specification version 3.0.4.**
Object deletion (*ODELG*)	**The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 3.0.4 feature.**
Secure carrier (*CarG*)	**The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecode verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.**

Subjects are active components of the TOE that (essentially) act on the behalf of users. The users of the TOE include people or institutions (like the Applet Developer, the Card Issuer, the Controlling Authority), hardware (like the CAD where the card is inserted or the PCD) and software components (like the application packages installed on the card). Some of the users may just be aliases for other users. For instance, the Controlling Authority in charge of the bytecode verification of the applications may be just an alias for the card issuer.

Subjects (prefixed with an "S") are described in the following table:

Subject	Description
S.ADEL	**The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in §7.1.3.1.**
S.APPLET	**Any applet instance.**

Subject	Description
S.BCV	**The bytecode verifier (BCV), which acts on behalf of the Controlling Authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in §7.1.3.4.**
S.CAD	**The CAD represents off-card entity that communicates with the S.INSTALLER.**
S.INSTALLER	**The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.**
S.JCRE	**The runtime environment under which Java programs in a smart card are executed.**
S.JCVM	**The bytecode interpreter that enforces the firewall at runtime.**
S.LOCAL	**Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.**
S.MEMBER	**Any object's field, static field or array position.**
S.PACKAGE	**A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.**

Objects (prefixed with an "O") are described in the following table:

Object	Description
O.APPLET	**Any installed applet, its code and data.**
O.CODE_PKG	**The code of a package, including all linking information. On the Java Card platform, a package is the installation unit.**
O.JAVAOBJECT	**Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.**

Information (prefixed with an "I") is described in the following table:

Information	Description
I.APDU	**Any APDU sent to or from the card through the communication channel.**
I.DATA	**JCVM Reference Data: objectref addresses of APDU buffer, JCRE-owned instances of APDU class and byte array for install method.**

Security attributes linked to these subjects, objects and information are described in the following table with their values:

Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs.
Applet Selection Status	"Selected" or "Deselected".
Applet's version number	The version number of an applet (package) indicated in the export file.
Class	Identifies the implementation class of the remote object.
Context	Package AID or "Java Card RE".
Currently Active Context	Package AID or "Java Card RE".
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([JCVM], §4.5.2).
ExportedInfo	Boolean (indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies the remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package). The owner of a remote object is the applet instance that created the object.
Package AID	The AID of each package indicated in the export file.
Registered Applets	The set of AID of the applet instances registered on the card.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Selected Applet Context	Package AID or "None".
Sharing	Standards, SIO, Java Card RE entry point or global array.
Static References	Static fields of a package may contain references to objects. The Static References attribute records those references.

(*) Transient objects of type CLEAR_ON_RESET behave like persistent objects in that they can be accessed only when the Currently Active Context is the object's context.

Operations (prefixed with "OP") are described in the following table. Each operation has parameters given between brackets, among which there is the "accessed object", the first one, when applicable. Parameters may be seen as security attributes that are under the control of the subject performing the operation.

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	**Read/Write an array component.**
OP.CREATE(Sharing, LifeTime) (*)	**Creation of an object (new or makeTransient call).**
OP.DELETE_APPLET(O.APPLET,...)	**Delete an installed applet and its objects, either logically or physically.**
OP.DELETE_PCKG(O.CODE_PKG,...)	**Delete a package, either logically or physically.**
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	**Delete a package and its installed applets, either logically or physically.**
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	**Read/Write a field of an instance of a class in the Java programming language.**
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	**Invoke a virtual method (either on a class instance or an array object).**
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	**Invoke an interface method.**
OP.JAVA(...)	**Any access in the sense of [JCRE], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.**
OP.PUT(S1,S2,I)	**Transfer a piece of information I from S1 to S2.**
OP.THROW(O.JAVAOBJECT)	**Throwing of an object (athrow, see [JCRE], §6.2.8.7).**

Operation	Description
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	**Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).**

(*) For this operation, there is no accessed object. This rule enforces that shareable transient objects are not allowed. For instance, during the creation of an object, the JavaCardClass attribute's value is chosen by the creator.

7.1.1.1 CoreG_LC Security Functional Requirements

This group is focused on the main security policy of the Java Card System, known as the firewall.

7.1.1.1.1 Firewall Policy

FDP_ACC.2-FIREWALL Complete access control

FDP_ACC.2.1-FIREWALL The TSF shall enforce the **FIREWALL access control SFP** on **S.PACKAGE, S.JCRE, S.JCVM, O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP.

Refinement:

The operations involved in the policy are:

- OP.CREATE,
- OP.INVK_INTERFACE,
- OP.INVK_VIRTUAL,
- OP.JAVA,
- OP.THROW,
- OP.TYPE_ACCESS.

FDP_ACC.2.2-FIREWALL The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

Application Note:

It should be noticed that accessing array's components of a static array, and more generally fields and methods of static objects, is an access to the corresponding O.JAVAOBJECT.

The operations under the control of the Firewall policy include:

- Reading and writing an array position or an instance field
- Invoking a virtual method on a class instance or an array
- Invoking an interface method on a class instance
- Throwing a Java Card exception

- Comparing the class of a class instance or an array against a given class
- Allocating a new class instance or array

See [JCSPP] for a detailed description of the subject, objects and operations under the control of the Firewall policy.

FDP_ACF.1-FIREWALL Security attribute based access control

FDP_ACF.1.1-FIREWALL The TSF shall enforce the **FIREWALL access control SFP** to objects based on the following:

Subject / Object	Security attributes
S.PACKAGE	LC Selection Status
S.JCVM	Active Applets, Currently Active Context
S.JCRE	Selected Applet Context
O.JAVAOBJECT	Sharing, Context, LifeTime

.

FDP_ACF.1.2-FIREWALL The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **R.JAVA.1 ([JCRE22], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**
- **R.JAVA.2 ([JCRE22], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**
- **R.JAVA.3 ([JCRE22], §6.2.8.10): S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.**
- **R.JAVA.4 ([JCRE22], §6.2.8.6): S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following conditions applies:**
  - **a) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",**
  - **b) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Non-multiselectable", and either "Package**

AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute Active Applets.

- **R.JAVA.5: S.PACKAGE may perform OP.CREATE only if the value of the Sharing parameter is "Standard"**.

FDP_ACF.1.3-FIREWALL The TSF shall explicitly authorise access of subjects to objects based on the following additional rules:

- **1) The subject S.JCRE can freely perform OP.JAVA(") and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**
- **2) The only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a Java Card API method (through OP.INVK_INTERFACE or OP.INVK_VIRTUAL)**.

FDP_ACF.1.4-FIREWALL The TSF shall explicitly deny access of subjects to objects based on the following additional rules:

- **1) Any subject with OP.JAVA upon an O.JAVAOBJECT whose LifeTime attribute has value "CLEAR_ON_DESELECT" if O.JAVAOBJECT's Context attribute is not the same as the Selected Applet Context.**
- **2) Any subject attempting to create an object by the means of OP.CREATE and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the Selected Applet Context**.

Application Note:

FDP_ACF.1.4/FIREWALL:

- The deletion of applets may render some O.JAVAOBJECT inaccessible, and the Java Card RE may be in charge of this aspect. This can be done, for instance, by ensuring that references to objects belonging to a deleted application are considered as a null reference. Such a mechanism is implementation-dependent.

In the case of an array type, fields are components of the array ([JVM], §2.14, §2.7.7), as well as the length; the only methods of an array object are those inherited from the Object class.

The Sharing attribute defines four categories of objects:

- Standard ones, whose both fields and methods are under the firewall policy,
- Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,
- JCRE entry points (Temporary or Permanent), who have freely accessible methods but protected fields,
- Global arrays, having both unprotected fields (including components; refer to JavaCardClass discussion above) and methods.

When a new object is created, it is associated with the Currently Active Context. But the object is owned by the applet instance within the Currently Active Context when the object is instantiated ([JCRE22], §6.1.3). An object is owned by an applet instance, by the JCRE or by the package library where it has been defined (these latter objects can only be arrays that initialize static fields of packages).

([JCRE22], Glossary) Selected Applet Context. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's AID, the Java Card RE makes this applet the Selected Applet Context. The Java Card RE sends all APDU commands to the Selected Applet Context.

While the expression "Selected Applet Context" refers to a specific installed applet, the relevant aspect to the policy is the context (package AID) of the selected applet. In this policy, the "Selected Applet Context" is the AID of the selected package.

([JCRE22], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).

It should be noticed that the invocation of static methods (or access to a static field) is not considered by this policy, as there are no firewall rules. They have no effect on the active context as well and the "acting package" is not the one to which the static method belongs to in this case.

It should be noticed that the Java Card platform, version 2.2.x and version 3 Classic Edition, introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([JCVM22], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.

An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There can only be one currently selected applet instance at a given time. ([JCRE22], §4).

The [JCSPP] introduces a detailed notation for defining the attributes and access rules for the Firewall policy. The detailed version of the rules is not repeated here for the sake of conciseness, but the following paragraphs provides a short summary of it.

An application instance has the following security attributes:

- The *Active Context* to which the applet instance belongs. Two instances of an applet declared in the same Java Card package belong to the same context.
- The *Selected Applet Context*, stating whether the instance is currently selected for execution.
- The *Currently Active Context*, which states what is the instance that is currently executing a bytecode. This attribute could be rather considered as TSF data supporting the rules of the policy.
- The *Multiselectable* attribute indicates whether the applet instance may be selected on several logical channels at the same time
- The *ActiveApplets* attribute lists all the applet instances declared in the same Java Card package as this one that are currently selected on a logical channel.

A class instance or array has the following attributes:

- Its sharing type, which classifies them into the following categories:
  - Standard objects, whose both fields and methods are under the firewall policy,
  - Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,
  - JCRE entry point objects, who have freely accessible methods but protected fields,

  - Global arrays, having both freely access positions and methods.
- Its owner, which is the applet instance that allocated the class instance or array.
- The lifetime of the information it contains, which may be either cleared when the last active applet in the contexts of its owner is deselected, or when the card is reset.

The Firewall policy introduce the following rules to determine the access to an object:

- Rule 1: The following operations may be performed on an object that is a JCRE Entry Point:
  - Reading or writing its instance fields or array positions
  - Invoking a virtual or interface method
  - Throwing the object as an exception
  - Comparing the class of the object against a given one
- Rule 2: An applet instance may perform any of the operations under the control of the policy on a JCRE Entry Point object or a global array.
- Rule 3: An applet instance may perform any of the operations under the control of the policy on the persistent standard objects owned by an applet instance belonging to the same context.
- Rule 4: An applet instance may perform the following operations on a persistent object owned by an applet instance belonging to a different context:
  - Invoking an interface method
  - Comparing the class of the object against a given class
- Rule 5: Any applet instance may create standard persistent objects

In addition to the conditional rules above, the Firewall policy also introduces a special rule for explicit access:

- Rule 6: The JCRE may perform any of the operations on any persistent object, and on any object containing transient information to be cleared when the card is reset.

Finally, the Firewall policy enforces the following rule for explicit denial of access:

- Rule 7: No subject different from the currently selected applet is allowed to perform an operation on objects containing transient information to be cleared on deselection of the current applet. This rule concerns both the operation that creates an object and the operation that access an existing object.
- Rule 8: An applet instance cannot invoke an interface method on a class instance that is owned by an applet instance belonging to a different context when this latter applet instance is not multi-selectable and is currently active on another logical channel. This rule is introduced by the *Logical Channels* group of requirements of [JCSPP].

FDP_IFC.1-JCVM Subset information flow control

FDP_IFC.1.1-JCVM The TSF shall enforce the **JCVM information flow control SFP** on **S.JCVM, S.LOCAL, S.MEMBER, I.DATA and OP.PUT(S1, S2, I)**.

Application Note:

It should be noticed that references of temporary Java Card RE entry points, which cannot be stored in class variables, instance variables or array components, are transferred from the internal memory of the Java Card RE (TSF data) to some stack through specific APIs (Java

Card RE owned exceptions) or Java Card RE invoked methods (such as the process(APDU apdu)); these are causes of OP.PUT(S1,S2,I) operations as well.

The information under the control of the JCVM information flow policy is the JCRE Temporary Entry point objects. The policy prevents those objects from being transferred from temporary storage (the operand stack, the local variables of a method) to persistent ones (static or instance fields, array positions).

See [JCSPP] for a detailed description of the subject, objects and operations under the control of this policy.

FDP_IFF.1-JCVM Simple security attributes

FDP_IFF.1.1-JCVM The TSF shall enforce the **JCVM information flow control SFP** based on the following types of subject and information security attributes:

Subjects	Security attributes
S.JCVM	Currently Active Context

.

FDP_IFF.1.2-JCVM The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- **An operation OP.PUT(S1, S.MEMBER, I.DATA) is allowed if and only if the Currently Active Context is "Java Card RE";**
- **Other OP.PUT operations are allowed regardless of the Currently Active Context's value**.

FDP_IFF.1.3-JCVM The TSF shall enforce the **following additional information flow control rules: none**.

FDP_IFF.1.4-JCVM The TSF shall explicitly authorise an information flow based on the following rules: **list of additional capabilities: none**.

FDP_IFF.1.5-JCVM The TSF shall explicitly deny an information flow based on the following rules: **when one of the conditions in the element FDP_IFF.1.1-JCVM above is not satisfied**.

Application Note:

The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE22], §6.2.8.1-3).

It should be noticed that this policy essentially applies to the execution of bytecode. Native methods, the Java Card RE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements. The way the Java Card virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the

implementation of the stack frame, may transit through an internal register prior to being pushed on the stack of the invoker. The returned bytecode would cause more than one OP.PUT operation under this scheme.

FDP_RIP.1-OBJECTS Subset residual information protection

FDP_RIP.1.1-OBJECTS The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** the following objects: **class instances and arrays**.

Application Note:

The semantics of the Java programming language requires for any object field and array position to be initialized with default values when the resource is allocated [JVM], §2.5.1.

FMT_MSA.1-JCRE Management of security attributes

FMT_MSA.1.1-JCRE The TSF shall enforce the **FIREWALL access control SFP** to restrict the ability to **modify** the security attributes **Selected Applet Context** to **the Java Card RE**.

Application Note:

The modification of the Selected Applet Context should be performed in accordance with the rules given in [JCRE22], §4 and [JCVM22], §3.4.

In Java Card specifications, the JCRE appears as the subject in charge of modifying the currently selected applets, see the chapter *Logical Channels and Applet Selection* of [JCRE]. In a smart card compliant with GlobalPlatform's specifications, the OPEN should therefore be considered as being part of the JCRE, as it is the subject in charge or selecting which is the applet instance that will be selected for the current session.

FMT_MSA.1-JCVM Management of security attributes

FMT_MSA.1.1-JCVM The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to restrict the ability to **modify** the security attributes **Currently Active Context and Active Applets** to **the Java Card VM (S.JCVM)**.

Application Note:

The modification of the Currently Active Context should be performed in accordance with the rules given in [JCRE22], §4 and [JCVM22], §3.4.

FMT_MSA.2-FIREWALL_JCVM Secure security attributes

FMT_MSA.2.1-FIREWALL_JCVM The TSF shall ensure that only secure values are accepted for **all the security attributes of subjects and objects defined in the FIREWALL access control SFP and the JCVM information flow control SFP**.

Application Note:

The following rules are given as examples only. For instance, the last two rules are motivated by the fact that the Java Card API defines only transient arrays factory methods. Future versions may allow the creation of transient objects belonging to arbitrary classes; such evolution will naturally change the range of "secure values" for this component.

- The Context attribute of an O.JAVAOBJECT must correspond to that of an installed applet or be "Java Card RE".
- An O.JAVAOBJECT whose Sharing attribute is a Java Card RE entry point or a global array necessarily has "Java Card RE" as the value for its Context security attribute.
- An O.JAVAOBJECT whose Sharing attribute value is a global array necessarily has "array of primitive type" as a JavaCardClass security attribute's value.
- Any O.JAVAOBJECT whose Sharing attribute value is not "Standard" has a PERSISTENT-LifeTime attribute's value.
- Any O.JAVAOBJECT whose LifeTime attribute value is not PERSISTENT has an array type as JavaCardClass attribute's value.

FMT_MSA.3-FIREWALL Static attribute initialisation

FMT_MSA.3.1-FIREWALL The TSF shall enforce the **FIREWALL access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2-FIREWALL [Editorially Refined] The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

Application Note:

FMT_MSA.3.1/FIREWALL

- Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable (FMT_MSA.1/JCRE). At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE22], §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".
- The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT_MSA.3.2/FIREWALL

- The intent is that none of the identified roles has privileges with regard to the default values of the security attributes. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP. The operation shall fail anyway if the created object would have had security attributes whose value violates FMT_MSA.2.1/FIREWALL_JCVM.

FMT_MSA.3-JCVM Static attribute initialisation

FMT_MSA.3.1-JCVM The TSF shall enforce the **JCVM information flow control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2-JCVM [Editorially Refined] The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions:

- **modify the Currently Active Context, the Selected Applet Context and the Active Applets**.

FMT_SMF.1-FIREWALL Specification of Management Functions

FMT_SMF.1.1-FIREWALL The TSF shall be capable of performing the following management functions: **initializing the *Active Context* and the *Selected Applet Context* and performing a context switch on the former when an instance method is invoked**.

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles**:**

- **Java Card RE (JCRE),**
- **Java Card VM (JCVM)**.

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

7.1.1.1.2 Application Programming Interface

The following SFRs are related to the Java Card API.

The whole set of cryptographic algorithms is generally not implemented because of limited memory resources and/or limitations due to exportation. Therefore, the following requirements only apply to the implemented subset.

It should be noticed that the execution of the additional native code is not within the TSF. Nevertheless, access to API native methods from the Java Card System is controlled by TSF because there is no difference between native and interpreted methods in their interface or invocation mechanism.

7.1.1.1.2.1 Key Generation

The TOE shall support on-card generation of different types of cryptographic keys.

FCS_CKM.1-Key_generation Cryptographic key generation

FCS_CKM.1.1-Key_generation The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **[assignment: cryptographic key generation algorithm]** and specified cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignement: list of standards]**

Iteration	Algorithm	Key Size	Standard
-ALG-RSA	Generating RSA key components using a true random number generator and Miller-Rabin algorithm for testing key components primality	1536 to 2048 bits	Annex A of IEEE P1363-2000
-ALG-RSA-CRT	Generating RSA-CRT key components using a true random number generator and Miller-Rabin algorithm for testing key components primality	1536 to 2048 bits	Annex A of IEEE P1363-2000
-APP-EC	Elliptic Curves private keys respecting a given EC domain and curve	224, 256, 384, 512 or 521 bits	ISO/IEC 15946-1, ISO/IEC 15946-3 and BSI's TR-03110
-APP-DH	Generating Diffie-Hellman keys using a random number generator and perfoming a modular exponential according to a given domain	1536 to 2048 bits	PKCS#3
-APP-DH-PKV	Generating Diffie-Hellman keys using a random number generator and perfoming a modular exponential according to a given domain	1536 to 2048 bits	PKCS#3
-ALG-EC-FP	Generating ECDH / ECDSA keys with Brainpool curve or NIST curve (for length 521 bits)	224 to 521 bits	ISO/IEC 15946-1, ISO/IEC 15946-3 and BSI's TR-03110

.

7.1.1.1.2.2 Key Agreement

The TOE supports several key agreement mechanisms based on the Diffie-Hellman protocol. This protocol enables the smart card to agree with card host on a shared secret that can be used to derive session keys. The TOE is not expected to directly distribute the session key (in the sense of sending the key value to the host) but it exchanges information with the host that enable each party to derive a shared secret on its own side.

The following SFR specify several methods for generating a shared secret that provides a suitable input for deriving a triple DES session key to be used in the Diffie-Hellman protocol, for instance using the method described in §4.3.3 (3DESKDF) of ISO/IEC 15946-3. However, this method is not intended to be the only possible method that an applet may use to derive a triple DES key. Actually, the Java Card Technology does not provide any specific service for creating cryptography key values of DES type, it just provides the means to create secure containers to hold such values. The DES keys that the applet places in such containers are ultimately provided by the Card Administrator or generated by the applet itself. The method used to create or import them is the responsibility of the TOE environment, which is subject to the OSP.SECRETS organizational policy.

FCS_CKM.2-Key_Agreement Cryptographic key distribution

FCS_CKM.2.1-Key_Agreement The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **[assignment: cryptographic key distribution method]** that meets the following: **[assignement: list of standards]**

Iteration	Distribution Method	Standard
-APP-SVDP-DH-PKCS3	Diffie-Hellman key agreement based on a modulo arithmetic based cryptographic algorithm	PKCS#3
-APP-EC-SVDP-DH	Diffie-Hellman key agreement based on an elliptic curve cryptography algorithm	IEEE P1363
-APP-EC-SVDP-DH-PLAIN	Diffie-Hellman key agreement based on an elliptic curve cryptography algorithm	IEEE P1363
-APP-EC-SVDP-EG	Diffie-Hellman key agreement based on El Gamal algorithm	IEEE P1363
-APP-DH-EG	Diffie-Hellman key agreement based on El Gamal algorithm	ISO 15946-1, ISO 15946-3 and BSI's TR03110

.

7.1.1.1.2.3 Application Cryptography Services

The following requirements describe the cryptoghraphy services provided in the Java Card API.

FCS_COP.1-APP-SHA Cryptographic operation

FCS_COP.1.1-APP-SHA The TSF shall perform **computation of a hash value for application instance's data** in accordance with a specified cryptographic algorithm **(SHA-224, SHA-256, SHA-384 and SHA-512)** and cryptographic key sizes **none** that meet the following: **FIPS 180-2**.

FCS_RND.1-APP Quality metric for random numbers

FCS_RND.1.1-APP The TSF shall provide a mechanism to generate random numbers that meet **the *STANDARD* level specified in [ANSSI]**.

FCS_COP.1-Asymetric Cryptographic operation

FCS_COP.1.1-Asymetric The TSF shall perform **[assignment: list of cryptographic operations]** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithm]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignement: list of standards]**

Iteration	Operation	Algorithm	Key Size	Standard
-APP-RSA	Signature generation, signature verification, encryption and decryption on application instance's data	RSA with or without padding, RSA_SHA	multiples of 32 from 1536 bits and up to 2048 bits	PKCS#1.5, ISO9796
-APP-RSA-CRT	Signature generation, signature verification, encryption and decryption on application instance's data	RSA CRT	multiples of 32 from 1536 bits and up to 2048 bits	PKCS#1.5, ISO9796
-APP-ECDSA	Signature generation and verification	Elliptic Curve DSA (ECDSA)	224, 256, 384, 512 or 521 bits	ISO-15946-1 and ISO-15946-2

.

Application Note:

The SFRs FCS_COP.1-Asymetric-APP-RSA and FCS_COP.1-Asymetric-APP-ECDSA are already present in [ICST] (names have been changed in this security target: FCS_COP.1-IC/RSA, FCS_COP.1-IC/ECDSA). For jTop platform:

- the ECDSA key length are not the same as in [ICST]: from 192 bits in [ICST] and from 224 bits for jTop platform

- the RSA key length are not the same as in [ICST]: from 1024 to 4096 in [ICST] and from 1536 to 2048 for jTop platform
- the standard for RSA is not the same as in [ICST].

FCS_COP.1-Symetric Cryptographic operation

FCS_COP.1.1-Symetric The TSF shall perform **[assignment: list of cryptographic operations]** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithm]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignement: list of standards]**

Iteration	Operation	Algorithm	Key Size	Standard
-APP-CIPHER/DES	Encryption and decryption of application instance's data	Triple DES either in CBC or ECB mode and with or without padding	112 or 168 bits	FIPS PUB 46-3, FIPS PUB 81, ISO 9797, Java Card Application Programming Interface
-APP-SIGN/DES	Signature generation and verification of application instance's data	8-bytes long MAC using Triple DES in CBC mode and with or without padding	112 or 168 bits	FIPS PUB 46-3, ISO 9797, Java Card Application Programming Interface
-APP-CIPHER/AES	Encryption and decryption of application instance's data	AES with block size 128 in CBC or ECB mode and without padding input data	128, 192, or 256 bits	FIPS PUB 197
-APP-SIGN/AES	Signature generation and verification of application instance's data	16-bytes long MAC using AES with block size 128 in CBC mode and no padding on input data, 16-bytes AES CMAC	128, 192, or 256 bits	FIPS PUB 197, NIST SP 800-38B

.

Application Note:

The SFRs FCS_COP.1-Symetric-APP-CIPHER/AES and FCS_COP.1-Symetric-APP-CIPHER/DES are already present in [ICST] (names have been changed in this security target: FCS_COP.1-IC/DES, FCS_COP.1-IC/AES). For jTOP platform, the standard needed are not the same as in [ICST], thus they are added in the ST.

7.1.1.1.2.4 Miscellaneous

FCS_CKM.3-KL Cryptographic key access

FCS_CKM.3.1-KL The TSF shall perform **[assignment: type of cryptographic key access]** in accordance with a specified cryptographic key access method **[assignment: cryptographic key access method]** that meets the following: **[assignement: list of standards]**

Iteration	Key access	Access method	Standard
-SD	SD key loading and replacement	through a PUT KEY or STORE DATA command	[GPCS]
-JCS	key access	class Key API	[JCAPI22]

.

FCS_CKM.4-KD Cryptographic key destruction

FCS_CKM.4.1-KD The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **[information removed]** that meets the following: **[information removed]**.

FDP_RIP.1-ABORT Subset residual information protection

FDP_RIP.1.1-ABORT The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any reference to an object instance created during an aborted transaction**.

FDP_RIP.1-APDU Subset residual information protection

FDP_RIP.1.1-APDU The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** the following objects: **the APDU buffer**.

Application Note:

The allocation of a resource to the APDU buffer is typically performed as the result of a call to the process() method of an applet.

FDP_RIP.1-bArray Subset residual information protection

FDP_RIP.1.1-bArray The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the bArray object**.

Application Note:

The `bArray` object is the byte array passed as real argument of the `Applet.install()` method when a new applet instance is created.

A resource is allocated to the bArray object when a call to an applet's install() method is performed. There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism (FDP_ROL.1.2/FIREWALL): the scope of the rollback does not extend outside the execution of the install() method, and the de-allocation occurs precisely right after the return of it.

FDP_RIP.1-KEYS Subset residual information protection

FDP_RIP.1.1-KEYS The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the cryptographic buffer (D.CRYPTO)**.

Application Note:

- The javacard.security & javacardx.crypto packages do provide secure interfaces to the cryptographic buffer in a transparent way. See javacard.security.KeyBuilder and Key interface of [JCAPI].

FDP_RIP.1-TRANSIENT Subset residual information protection

FDP_RIP.1.1-TRANSIENT The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any transient object**.

Application Note:

- The events that provoke the de-allocation of any transient object are described in [JCRE], §5.1.
- The clearing of CLEAR_ON_DESELECT objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable applet instances, CLEAR_ON_DESELECT memory segments may be attached to applets that are active in different logical channels. Multiselectable applet instances within a same package must share the transient memory segment if they are concurrently active ([JCRE], §4.2.

FDP_ROL.1-FIREWALL Basic rollback

FDP_ROL.1.1-FIREWALL The TSF shall enforce **the FIREWALL access control SFP and the JCVM information flow control SFP** to permit the rollback of the **operations OP.JAVA and OP.CREATE** on the **objects O.JAVAOBJECT**.

FDP_ROL.1.2-FIREWALL The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process(), install() or uninstall() call, notwithstanding the restrictions given in [JCRE22], §7.7, within the bounds of the Commit Capacity ([JCRE], §7.8), and those described in [JCAPI22]**.

Application Note:

Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism is either implemented in Java Card platform or relies on the transaction mechanism offered by the underlying platform. Some operations of the API are not conditionally updated, as documented in [JCAPI] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

7.1.1.1.3 Card Security Management

The following requirements are related to the security of the whole card, in contrast to the previous ones, that are somewhat restricted to the features of the Runtime Environment alone. For instance, a potential security violation detected by the virtual machine may require a reaction that does not only concern the virtual machine, such as requesting the appropriate security module with the power to block the card to perform the operation.

FAU_ARP.1-JCS Security alarms

FAU_ARP.1.1-JCS The TSF shall take **one of the following actions:**

- **throw an exception,**
- **lock the card session,**
- **reinitialize the Java Card System and its data,**
- **temporary disabling the services of the card until a privileged role performs a special action;**
- **definitely disabling all the services of the card**

upon detection of a potential security violation.

Refinement:

The "potential security violation" stands for one of the following events:

- CAP file inconsistency,
- typing error in the operands of a bytecode,
- applet life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context, (see abortTransaction(), [JCAPI] and ([JCRE], §7.6.2)
- violation of the Firewall or JCVM SFPs,

- unavailability of resources,
- array overflow,

Application Note:

- The developer shall provide the exhaustive list of actual potential security violations the TOE reacts to. For instance, other runtime errors related to applet's failure like uncaught exceptions.
- The bytecode verification defines a large set of rules used to detect a "potential security violation". The actual monitoring of these "events" within the TOE only makes sense when the bytecode verification is performed on-card.
- Depending on the context of use and the required security level, there are cases where the card manager and the TOE must work in cooperation to detect and appropriately react in case of potential security violation. This behavior must be described in this component. It shall detail the nature of the feedback information provided to the card manager (like the identity of the offending application) and the conditions under which the feedback will occur (any occurrence of the java.lang.SecurityException exception).
- The "locking of the card session" may not appear in the policy of the card manager. Such measure should only be taken in case of severe violation detection; the same holds for the re-initialization of the Java Card System. Moreover, the locking should occur when "clean" re-initialization seems to be impossible.
- The locking may be implemented at the level of the Java Card System as a denial of service (through some systematic "fatal error" message or return value) that lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager). Finally, because the installation of applets is a sensitive process, security alerts in this case should also be carefully considered herein.

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors on cryptographic keys and PIN values** on all objects, based on the following attributes: **a checksum on the value of those objects**.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **mute the card if an application attempts to use the corrupted key or PIN**.

Application Note:

- Although no such requirement is mandatory in the Java Card specification, at least an exception shall be raised upon integrity errors detection on cryptographic keys, PIN values and their associated security attributes. Even if all the objects cannot be monitored, cryptographic keys and PIN objects shall be considered with particular attention by ST authors as they play a key role in the overall security.
- It is also recommended to monitor integrity errors in the code of the native applications and Java Card applets.
- For integrity sensitive application, their data shall be monitored (D.APP_I_DATA): applications may need to protect information against unexpected modifications, and explicitly control whether a piece of information has been changed between two accesses. For example, maintaining the integrity of an electronic purse's balance is extremely important because this value represents real money. Its modification must

be controlled, for illegal ones would denote an important failure of the payment system.

- A dedicated library could be implemented and made available to developers to achieve better security for specific objects, following the same pattern that already exists in cryptographic APIs, for instance.

FPR_UNO.1-CRYPTO Unobservability

FPR_UNO.1.1-CRYPTO The TSF shall ensure that **external users or malicious application instances fraudulently installed on the card** are unable to observe the operation **encryption, decryption, signature generation and verification** on **application instance's data** by **application instances**.

Application Note:

The SD shall be considered as a distinguished application instance which is therefore covered by this requirement.

FPR_UNO.1-PIN Unobservability

FPR_UNO.1.1-PIN The TSF shall ensure that **external users or malicious application instances fraudulently installed on the card** are unable to observe the operation **comparison** on **PIN codes** by **application instances**.

FPT_FLS.1-JCS Failure with preservation of secure state

FPT_FLS.1.1-JCS The TSF shall preserve a secure state when the following types of failures occur: **those associated to the potential security violations described in FAU_ARP.1-JCS**.

FPT_TDC.1 Inter-TSF basic TSF data consistency

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret **the CAP files, the bytecode and its data arguments** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use

- **the rules defined in [JCVM] specification,**
- **the API tokens defined in the export files of reference implementation,**
- **The ISO 7816-6 rules**

when interpreting the TSF data from another trusted IT product.

Application Note:

Concerning the interpretation of data between the TOE and the underlying Java Card platform, it is assumed that the TOE is developed consistently with the SCP functions, including memory management, I/O functions and cryptographic functions.

7.1.1.1.4 AID Management

FIA_ATD.1-AID User attribute definition

FIA_ATD.1.1-AID The TSF shall maintain the following list of security attributes belonging to individual users:

- **Package AID,**
- **Applet's version number,**
- **Registered applet AID,**
- **Applet Selection Status ([JCVM], §6.5)**.

Refinement:

"Individual users" stand for applets.

FIA_UID.2-AID User identification before any action

FIA_UID.2.1-AID The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application Note:

- By users here it must be understood the ones associated to the packages (or applets) that act as subjects of policies. In the Java Card System, every action is always performed by an identified user interpreted here as the currently selected applet or the package that is the subject's owner. Means of identification are provided during the loading procedure of the package and the registration of applet instances.
- The role Java Card RE defined in FMT_SMR.1 is attached to an IT security function rather than to a "user" of the CC terminology. The Java Card RE does not "identify" itself to the TOE, but it is part of it.

This requirement refers to the identification of the application instances when they request a service from the TOE that is under the control of one of the TSP defined in this Security Target. In particular, the access control rules of the Firewall and the Cardholder Verification Method requires identifying the AID of the application instance that requires access to a Java Card object or to the global PIN of the card.

FIA_USB.1-AID User-subject binding

FIA_USB.1.1-AID The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: **Package AID**.

FIA_USB.1.2-AID The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **when an instance of an applet class declared in a Java Card package P is created, that package P is taken as the active context associated to the new application instance**.

FIA_USB.1.3-AID The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **none**.

Application Note:

The user is the applet and the subject is the S.PACKAGE. The subject security attribute "Context" shall hold the user security attribute "package AID".

FMT_MTD.1-JCRE Management of TSF data

FMT_MTD.1.1-JCRE The TSF shall restrict the ability to **modify** the **list of registered applets AIDs** to **the JCRE**.

Application Note:

- The installer and the Java Card RE manage some other TSF data such as the applet life cycle or CAP files, but this management is implementation specific. Objects in the Java programming language may also try to query AIDs of installed applets through the lookupAID(...) API method.
- The installer, applet deletion manager or even the card manager may be granted the right to modify the list of registered applets' AIDs in specific implementations (possibly needed for installation and deletion; see #.DELETION and #.INSTALL).

FMT_MTD.3-AID Secure TSF data

FMT_MTD.3.1-AID The TSF shall ensure that only secure values are accepted for **the registered applets' AIDs**.

Application Note:

This SFR corresponds to FMT_MTD.3/JCRE of [JCSPP].

This requirement concerns the use of valid Application IDentifiers (AID). An AID is valid if it is unique, has the right length, and was specified in the INSTALL command that created the Executable File or applet instance that it identifies.

7.1.1.2 InstG Security Functional Requirements

This group consists of the SFRs related to the installation of the applets, which addresses security aspects outside the runtime. The installation of applets is a critical phase, which lies partially out of the boundaries of the firewall, and therefore requires specific treatment. In this PP, loading a package or installing an applet modeled as importation of user data (that is, user application's data) with its security attributes (such as the parameters of the applet used in the firewall rules).

FDP_ITC.2-CCM Import of user data with security attributes

FDP_ITC.2.1-CCM The TSF shall enforce the **PACKAGE LOADING information flow control SFP and the Secure Channel Protocol information flow policy** when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2-CCM The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3-CCM The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4-CCM The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5-CCM The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE:

- **An Executable Load File may depend on (import or use data from) other Executable Files already installed on the card. This dependency is explicitly stated in the Executable Load File in the form of a list of Executable File AIDs. The loading is allowed only if, for each dependent Executable File, its AID attribute is equal to a resident Executable File AID attribute, and the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM], §4.5.2). The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM], §4.4)**.

Application Note:

This SFR corresponds to FDP_ITC.2/Installer of [JCSPP].

FDP_ITC.2.1-CCM:

- The most common importation of user data is package loading and applet installation on the behalf of the installer. Security attributes consist of the shareable flag of the class component, AID and version numbers of the package, maximal operand stack size and number of local variables for each method, and export and import components (accessibility).

FDP_ITC.2.3-CCM:

- The format of the CAP file is precisely defined in [JCVM22] specifications; it contains the user data (like applet's code and data) and the security attributes altogether. Therefore there is no association to be carried out elsewhere.

FDP_ITC.2.4-CCM:

- Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM22], §4.5). With the AID, it describes the package defined in the CAP file. When an export file is used during preparation of a CAP file, the versions numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM22], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatible. However, package files do have "package Version Numbers" ([JCVM22]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

FDP_ITC.2.5-CCM:

- A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs.
- The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM22], §4.4).
- The installation (the invocation of an applet's install method by the installer) is implementation dependent ([JCRE22], §11.2).
- Other rules governing the installation of an applet, that is, its registration to make it SELECTable by giving it a unique AID, are also implementation dependent (see, for example, [JCRE22], §11).

FMT_SMR.1-CCM Security roles

FMT_SMR.1.1-CCM The TSF shall maintain the roles**: Installer**.

FMT_SMR.1.2-CCM The TSF shall be able to associate users with roles.

Application Note:

This SFR corresponds to FMT_SMR.1/Installer of [JCSPP].

FPT_FLS.1-CCM Failure with preservation of secure state

FPT_FLS.1.1-CCM The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in [JCRE22] §11.1.45**.

Application Note:

This SFR corresponds to FPT_FLS.1/Installer of [JCSPP].

FPT_RCV.3-CCM/ELF Automated recovery without undue loss

FPT_RCV.3.1-CCM/ELF When automated recovery from **a failure or service discontinuity** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2-CCM/ELF For **abortion of the installation process of an Executable Load File, detection of a potential loss of integrity during the transmission of an Executable Load File to the card, and any fatal error occurred during the linking of an Executable Load File to the Executable Files already installed on the card**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3-CCM/ELF The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **the loss of the Executable Load File being installed** for loss of TSF data or objects under the control of the TSF.

FPT_RCV.3.4-CCM/ELF The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.

Application Note:

This SFR corresponds to FPT_RCV.3/Installer of [JCSPP].

FPT_RCV.3.1-CCM/ELF:

- This element is not within the scope of the Java Card specification, which only mandates the behavior of the Java Card System in good working order. Further details on the "maintenance mode" shall be provided in specific implementations. The following is an excerpt from [CC2], p298: In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorised users should be allowed access to this mode but the real details of who can access this mode is a function of FMT: Security management. If FMT: Security management does not put any controls on who can access this mode, then it may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the SFRs.

FPT_RCV.3.2-CCM/ELF:

- Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE22], §11.1.5 for possible scenarios. Precise behavior is left to implementers. This component shall include among the listed failures the deletion of a package/applet. See ([JCRE22], 11.3.4) for possible scenarios. Precise behavior is left to implementers.
- Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [PP0035]) and, from the TOE's side, by events "that clear transient objects" and transactional features. See FPT_FLS.1-IC, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT and FDP_ROL.1/FIREWALL.

FPT_RCV.3.3-CCM/ELF:

- The quantification is implementation dependent, but some facts can be recalled here. First, the SCP ensures the atomicity of updates for fields and objects, and a power-failure during a transaction or the normal runtime does not create the loss of otherwise-permanent data, in the sense that memory on a smart card is essentially persistent with this respect (EEPROM). Data stored on the RAM and subject to such failure is intended to have a limited lifetime anyway (runtime data on the stack, transient objects' contents). According to this, the loss of data within the TSF scope should be limited to the same restrictions of the transaction mechanism.

FPT_RCV.3-CCM/AI Automated recovery without undue loss

FPT_RCV.3.1-CCM/AI When automated recovery from **a failure or service discontinuity** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2-CCM/AI For **abortion or any error or exception thrown during the installation process of a new application instance**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3-CCM/AI The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **the loss of the Java Card objects created during the installation of the new Application instance** for loss of TSF data or objects under the control of the TSF.

FPT_RCV.3.4-CCM/AI The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.

Application Note:

This SFR corresponds to FPT_RCV.3/Installer of [JCSPP].

FPT_RCV.3.1-CCM/AI:

- This element is not within the scope of the Java Card specification, which only mandates the behavior of the Java Card System in good working order. Further details on the "maintenance mode" shall be provided in specific implementations. The following is an excerpt from [CC2], p298: In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorised users should be allowed access to this mode but the real details of who can access this mode is a function of FMT: Security management. If FMT: Security management does not put any controls on who can access this mode, then it may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the SFRs.

FPT_RCV.3.2-CCM/AI:

- Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE22], §11.1.5 for possible scenarios. Precise behavior is left to implementers. This component shall include among the listed failures the deletion of a

package/applet. See ([JCRE22], 11.3.4) for possible scenarios. Precise behavior is left to implementers.

- Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [PP0035]) and, from the TOE's side, by events "that clear transient objects" and transactional features. See FPT_FLS.1-IC, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT and FDP_ROL.1/FIREWALL.

FPT_RCV.3.3-CCM/AI:

- The quantification is implementation dependent, but some facts can be recalled here. First, the SCP ensures the atomicity of updates for fields and objects, and a power-failure during a transaction or the normal runtime does not create the loss of otherwise-permanent data, in the sense that memory on a smart card is essentially persistent with this respect (EEPROM). Data stored on the RAM and subject to such failure is intended to have a limited lifetime anyway (runtime data on the stack, transient objects' contents). According to this, the loss of data within the TSF scope should be limited to the same restrictions of the transaction mechanism.

7.1.1.3 ADELG Security Functional Requirements

This group consists of the SFRs related to the deletion of applets and/or packages, enforcing the applet deletion manager (ADEL) policy on security aspects outside the runtime. Deletion is a critical operation and therefore requires specific treatment. This policy is better thought as a frame to be filled by ST implementers.

FDP_ACC.2/ADEL Complete access control

FDP_ACC.2.1/ADEL The TSF shall enforce the **ADEL access control SFP** on **S.ADEL, S.JCRE, S.JCVM, O.JAVAOBJECT, O.APPLET and O.CODE_PKG** and all operations among subjects and objects covered by the SFP.

Refinement:

The operations involved in the policy are:

- OP.DELETE_APPLET,
- OP.DELETE_PCKG,
- OP.DELETE_PCKG_APPLET.

FDP_ACC.2.2/ADEL The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

Application Note:

The operations under the control of the ADEL policy include:

- Deleting an applet instance installed on the card;
- Deleting an Executable File loaded on the card.

In a smart card compliant with [GPCS], the role of the *Applet Deletion Manager* is played by the Issuer Security Domain and the Delegated Management, and the deletion operations are performed through the DELETE APDU command. See [JCSPP] for a detailed description of the subject, objects and operations under the control of this policy.

FDP_ACF.1/ADEL Security attribute based access control

FDP_ACF.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to objects based on the following:

Subject/Object	Attributes
S.JCVM	Active Applets
S.JCRE	Selected Applet Context, Registered Applets, Resident Packages
O.CODE_PKG	Package AID, Dependent Package AID, Static References
O.APPLET	Applet Selection Status
O.JAVAOBJECT	Owner, Remote

.

FDP_ACF.1.2/ADEL The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

In the context of this policy, an object O is reachable if and only one of the following conditions hold:

- **(1) the owner of O is a registered applet instance A (O is reachable from A),**
- **(2) a static field of a resident package P contains a reference to O (O is reachable from P),**
- **(3) there exists a valid remote reference to O (O is remote reachable),**
- **(4) there exists an object O' that is reachable according to either (1) or (2) or (3) above and O' contains a reference to O (the reachability status of O is that of O').**

The following access control rules determine when an operation among controlled subjects and objects is allowed by the policy:

- **R.JAVA.14 ([JCRE22], §11.3.4.1, Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) there is no instance in the context of O.APPLET that is active in any logical channel and**
  - **(3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote reachable.**
- **R.JAVA.15 ([JCRE22], §11.3.4.1, Multiple Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) there is no instance of any of the O.APPLET being deleted that is active in any logical channel and**

- **(3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote reachable.**

o **R.JAVA.16 ([JCRE22], §11.3.4.2, Applet/Library Package Deletion): S.ADEL may perform OP.DELETE_PCKG upon an O.CODE_PKG only if,**

- **(1) S.ADEL is currently selected,**
- **(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG that is an instance of a class that belongs to O.CODE_PKG, exists on the card and**
- **(3) there is no resident package on the card that depends on O.CODE_PKG.**

o **R.JAVA.17 ([JCRE22], §11.3.4.3, Applet Package and Contained Instances Deletion): S.ADEL may perform OP.DELETE_PKG_APPLET upon an O.CODE_PKG only if,**

- **(1) S.ADEL is currently selected,**
- **(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG, which is an instance of a class that belongs to O.CODE_PKG exists on the card,**
- **(3) there is no package loaded on the card that depends on O.CODE_PKG, and**
- **(4) for every O.APPLET of those being deleted it holds that: (i) there is no instance in the context of O.APPLET that is active in any logical channel and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE22], §8.5) O.JAVAOBJECT is remote reachable**.

FDP_ACF.1.3/ADEL The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4/ADEL [Editorially Refined] The TSF shall explicitly deny access of **any subject but S.ADEL to O.CODE_PKG or O.APPLET for the purpose of deleting them from the card**.

Application Note:

FDP_ACF.1.2/ADEL:

- This policy introduces the notion of reachability, which provides a general means to describe objects that are referenced from a certain applet instance or package.
- S.ADEL calls the "uninstall" method of the applet instance to be deleted, if implemented by the applet, to inform it of the deletion request. The order in which these calls and the dependencies checks are performed are out of the scope of this protection profile.

The [JCSPP] introduces a detailed notation for defining the attributes and access rules for the ADEL policy. The detailed version of the rules is not repeated here for the sake of conciseness, but the following paragraphs provide a short summary of it.

An application instance has the following security attributes:

- The *Selection State* specifies whether the applet is currently active on some logical channel.

A class instance or array has the following security attributes:

- The *Owner* specifies the applet instance that created the class instance or array;
- The *Class* specifies of which class the object is an instance of. This attribute is not explicitly mentioned in [JCSPP] but is implicit in the access control rules of the ADEL policy.

An Executable File has the following security attributes:

- The *Name* identifies the file;
- The *Imported Files* specifies the files on which it depends;
- The *Static References* specifies the class instances or arrays that are directly reachable from the file.

The ADEL policy introduce the following rules to delete an applet instance or Executable File:

- Rule ADEL-1: An applet instance may only be deleted by the Applet Deletion Manager, provided that the applet instance is not currently active, and that all its class instances and arrays are neither reachable from an Executable File (different from the one declaring the applet, according to the refinement made by ths ST), nor from other applet instances.
- Rule ADEL-2: An Executable File containing a library may only be deleted by the Applet Deletion Manager, provided that no other file depends on this one, and that there is no instance of a class defined in this file that is reachable from other files.
- Rule ADEL-3: An Executable File declaring Applet classes may only be deleted by the Applet Deletion Manager, provided that it fulfills the premises in Rule 2, and that all the instances of the Applets declared in it satisfy the premises in Rule 1.

The ADEL policy in [JCSPP] contains an extra rule concerning the deletion of a collection of applet instances in a single step (rule R.JAVA.15, *"Multiple Applet Instance Deletion"*). That rule does not apply to a TOE compliant with GlobalPlatform, as this standard does not support multiple applet deletion.

Finally, the ADEL policy explicitly denies deletion access to other subjects different from the Applet Deletion Manager.

FDP_RIP.1/ADEL Subset residual information protection

FDP_RIP.1.1/ADEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**.

Application Note:

Deleted freed resources (both code and data) may be reused, depending on the way they were deleted (logically or physically). Requirements on de-allocation during applet/package deletion are described in [JCRE22], §11.3.4.1, §11.3.4.2 and §11.3.4.3.

FMT_MSA.1/ADEL Management of security attributes

FMT_MSA.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to restrict the ability to **modify** the security attributes **Registered Applets and Resident Packages** to **the Java Card RE**.

FMT_MSA.3/ADEL Static attribute initialisation

FMT_MSA.3.1/ADEL The TSF shall enforce the **ADEL access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/ADEL The TSF shall allow the **following role(s): none,** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMF.1/ADEL Specification of Management Functions

FMT_SMF.1.1/ADEL The TSF shall be capable of performing the following management functions: **modify the list of registered applets' AIDs and the Resident Packages**.

FMT_SMR.1/ADEL Security roles

FMT_SMR.1.1/ADEL The TSF shall maintain the roles**: applet deletion manager**.

FMT_SMR.1.2/ADEL The TSF shall be able to associate users with roles.

Application Note:

In a smart card compliant with [GPCS], the role of the *Applet Deletion Manager* is played by the Issuer Security Domain and the Delegated Management Security Domain.

FPT_FLS.1/ADEL Failure with preservation of secure state

FPT_FLS.1.1/ADEL The TSF shall preserve a secure state when the following types of failures occur: **the applet deletion manager fails to delete a package/applet as described in [JCRE22], §11.3.4**.

Application Note:

- The TOE may provide additional feedback information to the card manager in case of a potential security violation (see FAU_ARP.1).
- The Package/applet instance deletion must be atomic. The "secure state" referred to in the requirement must comply with Java Card specification ([JCRE22], §11.3.4.)

7.1.1.4 ODELG Security Functional Requirements

The following requirements concern the object deletion mechanism. This mechanism is triggered by the applet that owns the deleted objects by invoking a specific API method.

FDP_RIP.1/ODEL Subset residual information protection

FDP_RIP.1.1/ODEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the objects owned by the context of an applet instance which triggered the execution of the method `javacard.framework.JCSystem.requestObjectDeletion()`**.

Application Note:

- Freed data resources resulting from the invocation of the method javacard.framework.JCSystem.requestObjectDeletion() may be reused. Requirements on de-allocation after the invocation of the method are described in [JCAPI22].
- There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism: the execution of requestObjectDeletion() is not in the scope of the rollback because it must be performed in between APDU command processing, and therefore no transaction can be in progress.

FPT_FLS.1/ODEL Failure with preservation of secure state

FPT_FLS.1.1/ODEL The TSF shall preserve a secure state when the following types of failures occur: **the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the `JCSystem.requestObjectDeletion()` method**.

Application Note:

The TOE may provide additional feedback information to the card manager in case of potential security violation (see FAU_ARP.1).

7.1.1.5 CarG Security Functional Requirements

This group includes requirements for preventing the installation of packages that has not been bytecode verified, or that has been modified after bytecode verification.

FCO_NRO.2-TOKEN Enforced proof of origin

FCO_NRO.2.1-TOKEN The TSF shall enforce the generation of evidence of origin for transmitted **card content management operation requests processed by a Security Domain with Delegated Management privilege** at all times.

FCO_NRO.2.2-TOKEN The TSF shall be able to relate the **token present in the card content management operation request** of the originator of the information, and the **parameters of the card content management operation request (as defined in Section C.4 of [GPCS])** of the information to which the evidence applies.

FCO_NRO.2.3-TOKEN The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **none**.

FCO_NRO.2-DAP Enforced proof of origin

FCO_NRO.2.1-DAP The TSF shall enforce the generation of evidence of origin for transmitted **application packages** at all times.

FCO_NRO.2.2-DAP The TSF shall be able to relate the **identity** of the originator of the information, and the **Executable Load Files** of the information to which the evidence applies.

FCO_NRO.2.3-DAP The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **immediate verification**.

Application Note:

FCO_NRO.2-SC: The FCO_NRO.2/CM SFR of the [JCSPP] has been renamed into this FCO_NRO.2-DAP SFR to convey the idea of the DAP mechanism used by GlobalPlatform for this purpose.

FCO_NRO.2.1-SC:

- Upon reception of a new application package for installation, the card manager shall first check that it actually comes from the Controlling Authority (Mandated DAP Verification functionality). The Controlling Authority is the entity responsible for bytecode verification.
- Upon reception of a new application package for installation, if requested so by the Application Provider, the card manager shall check that the new application package actually comes from the Application Provider (DAP Verification functionality).

FCO_NRO.2.3-SC:

- The card manager performs an immediate verification of the origin of the package using an electronic signature mechanism (DAP Verification), and no evidence is kept on the card for future verifications.

FDP_IFC.2-SCP Complete information flow control

FDP_IFC.2.1-SCP The TSF shall enforce the **PACKAGE LOADING information flow control SFP** on **S.INSTALLER, S.BCV, S.CAD and I.APDU** and all operations that cause that information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2-SCP The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

Refinement:

- The subjects covered by this policy are those involved in the exchange of messages between the card and the CAD through a potentially unsafe communication channel:
  - An off-card subject that represents the Card Administrator (S.BCV).
  - Any application with the Security Domain privilege (S.CRD).
  - Any other subject that may potentially insert, delete, modify, or permute the messages exchanged between the two former subjects (S.SPY).
- The information controlled by this policy is the one contained in the APDU commands sent to the card and their associated responses returned to the CAD. This includes Executable Files, file and application names, application privileges, application and card life-cycle states, smart card and Card Issuer identification data, the Security Domain keys, personalization data, and the requests to and the responses from the services offered by the embedded applications

Application Note:

This SFR corresponds to FDP_IFC.2/CM of [JCSPP].

- The subjects covered by this policy are those involved in the loading of an application package by the card through a potentially unsafe communication channel.
- The operations that make information to flow between the subjects are those enabling to send a message through and to receive a message from the communication channel linking the card to the outside world. It is assumed that any message sent through the channel as clear text can be read by an attacker. Moreover, an attacker may capture any message sent through the communication channel and send its own messages to the other subjects.
- The information controlled by the policy is the APDUs exchanged by the subjects through the communication channel linking the card and the CAD. Each of those messages contain part of an application package that is required to be loaded on the card, as well as any control information used by the subjects in the communication protocol.

The operations that make information to flow between the subjects are to send a message through and receive a message from the communication channel linking the card to the outside world. Such messages are the GlobalPlatform commands and their associated responses, which are schematically characterized as follows:

- SEND(M): A subject sends a message M through the communication channel.
- RECEIVE(M): A subject receives a message M from the communication channel.

The subjects controlled by this security policy are the same as in [JCSPP], but they have been renamed according to GlobalPlatform's terminology:

- The off-card entity responsible for the bytecode verification of the Executable Load File, which is named S.BCV in [JCSPP], is called Card Administrator in this Security Target.
- The on-card entity responsible for the downloading of the Load File, which is called S.CRD in [JCSPP], is called the Security Domain in this Security Target.
- The attacker, called S.SPY in [JCSPP], is also called Spy in this Security Target.

FDP_IFF.1-SCP Simple security attributes

FDP_IFF.1.1-SCP The TSF shall enforce the **Secure Channel Protocol information flow control policy (SCP)** based on the following types of subject and information security attributes:

- **The messages exchanged between the on-card and the off-card subjects have a single security attribute, namely, the *MAC* ensuring the integrity and the origin of the message**
- **The on-card and the off-card subjects have the following security attributes:**
  - **The *Challenge* is a random number generated by the subject in order to identify the current session.**
  - **The *Cryptogram* is a secret relative to the current smart card session that serves to authenticate the on- and off-card subjects (but not the spy). The cryptogram is derived from the challenges of both the card and the terminal.**
  - **The *Key Set* is a collection of key triplets (called key set) used to encrypt the Derivation Data in order to generate the session keys. Each key set is composed of a Secure Channel Encryption Key (S-ENC), a Command Message Authentication Code Key (C-MAC) and a Data Encryption Key (DEK).**
  - **The *Static Keys* is a collection of key sets, each one identified by a key version number.**
  - **The *Session Keys* is a set of keys used to verify the origin and integrity of the received message, and to decrypt their contents. This set is made of the following keys:**
    - **Command Message Authentication Code Key (C-MAC session key);**
    - **Encryption Key (S-ENC session key);**
    - **Data Encryption Key (DEK session key).**
  - **The *Sequence Counter* is a counter attached to each Key Set, which is used to derive the session keys.**
  - **The *Initial Chaining Vector (ICV)* is a value used to compute the MAC value of a message, which relates it to the previous messages of the current session.**
- **In addition to the abovementioned ones, the SD have an extra attribute, namely, the *Command Security Level* defined for the messages that the card receives through the secure channel. The possible security levels are: NO-SEC (clear text), C-AUTHENTICATED (authentication of the command's issuer), C-MAC (authentication of the issuer and integrity of**

the command), C-DEC (authentication of the issuer, integrity and confidentiality of the command).

FDP_IFF.1.2-SCP The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- **The SD may process a RECEIVE(INITIALIZE-UPDATE) operation only if the key set specified in the command exist among the static keys of the SD.**
- **The SD may process a RECEIVE(EXTERNAL-AUTHENTICATE) operation if the following conditions hold:**
  - **The cryptogram received from the off-card subject is equal to the cryptogram computed by the Security Domain.**
  - **The MAC attached to the message has been generated from the C-MAC session key and the current value of the ICV.**
- **The SD may process a RECEIVE (GET-DATA) operation if the following condition holds:**
  - **If the command security level is at least C-MAC, the MAC attached to the message has been generated from the command using the C-MAC session key and the current value of the ICV.**
- **The SD may process a RECEIVE (M) operation for any other command M different from the ones cited in the rules above if the following conditions hold:**
  - **The current security level is at least AUTHENTICATED.**
  - **If the command security level is at least C-MAC, the MAC attached to the message has been generated from the clear-text command using the C-MAC session key and the current value of the ICV**.

FDP_IFF.1.3-SCP The TSF shall enforce the **following additional information flow control SFP rules: none**.

FDP_IFF.1.4-SCP The TSF shall explicitly authorise an information flow based on the following rules: **list of additional SFP capabilities: none**.

FDP_IFF.1.5-SCP The TSF shall explicitly deny an information flow based on the following rules: **when none of the conditions listed in the element FDP_IFF.1.4 of this component hold and at least one of those listed in the element FDP_IFF.1.2 does not hold**.

Application Note:

This SFR corresponds to FDP_IFF.1/CM of [JCSPP].

The flow of messages in the rules of the policy is described from the card's side: when the operation is SEND, the subject sending the message is the on-card one, when the operation is RECEIVE, the subject receiving the message is the on-card one.

FDP_IFF.1.1/CM:

- The security attributes used to enforce the PACKAGE LOADING SFP are implementation dependent. More precisely, they depend on the communication protocol enforced between the CAD and the card. For instance, some of the attributes that can be used

are: (1) the keys used by the subjects to encrypt/decrypt their messages; (2) the number of pieces the application package has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the package, etc. See for example Appendix D of [GP].

FDP_IFF.1.2/CM:

- The precise set of rules to be enforced by the function is implementation dependent. The whole exchange of messages shall verify at least the following two rules: (1) the subject S.INSTALLER shall accept a message only if it comes from the subject S.CAD; (2) the subject S.INSTALLER shall accept an application package only if it has received without modification and in the right order all the APDUs sent by the subject S.CAD.

FDP_IFF.1.5/CM:

- The verification of the integrity and authenticity evidences can be performed either during loading or during the first installation of an application of the package.

FDP_UIT.1-CCM Data exchange integrity

FDP_UIT.1.1-CCM The TSF shall enforce the **Secure Channel Protocol (SCP) information flow control policy, Security Domains (SD) access control policy and the PACKAGE LOADING information flow control SFP** to **receive** user data in a manner protected from **modification, deletion, insertion and replay** errors.

FDP_UIT.1.2-CCM The TSF shall be able to determine on receipt of user data, whether **modification, deletion, insertion and replay** has occurred.

Application Note:

This SFR corresponds to FDP_UIT.1/CM of [JCSPP].

Modification errors should be understood as modification, substitution, unrecoverable ordering change of data and any other integrity error that may cause the application package to be installed on the card to be different from the one sent by the CAD.

The user data to be protected are the new Executable Files received by the card.

This Security Target refines the FDP_UIT.1/CM requirement introduced in [JCSPP] by requesting the enforcement of the SD access control policy. Executable Files are sent as a sequence of LOAD commands. The SCP policy protects each single LOAD command from being modified during its transmission to the card. The SD policy prevents the LOAD commands sent to the card from being inserted, replayed or deleted, see Rule 6 of that policy.

FIA_UID.1-SC Timing of identification

FIA_UID.1.1-SC The TSF shall allow **the mediated actions listed below:**

- **selecting an application on the card;**

- **requesting data that identifies the card or the Card Issuer, provided that this feature has not been disabled during the Platform Initialization Phase;**
- **initializing a secure communication channel with the card**

on behalf of the user to be performed before the user is identified.

FIA_UID.1.2-SC The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application Note:

This SFR corresponds to FIA_UID.1/CM of [JCSPP].

The list of TSF-mediated actions is implementation-dependent, but package installation requires the user to be identified. Here by user is meant the one(s) that in the Security Target shall be associated to the role(s) defined in the component FMT_SMR.1/CM.

The actions that can be performed before identification in this security requirement are specific to the opening of a secure communication channel with the CAD. Other mediated actions regarding other identification process are listed in the component FIA_UID.2-AID.

FMT_MSA.1-SD.1 Management of security attributes

FMT_MSA.1.1-SD.1 The TSF shall enforce the **Security Domain access control policy (SD)** to restrict the ability to **modify** the security attributes ***Application State*** **of an application instance** to **the Issuer Security Domain, a Delegated Management Security Domain, the associated Security Domain, applications having the Global Lock privilege and the application instance itself**.

FMT_MSA.1-SD.2 Management of security attributes

FMT_MSA.1.1-SD.2 The TSF shall enforce the **Security Domain access control policy (SD)** to restrict the ability to **modify** the security attributes ***Card Reset*** to **the Issuer Security Domain**.

FMT_MSA.1-SD.3 Management of security attributes

FMT_MSA.1.1-SD.3 The TSF shall enforce the **Security Domain (SD) access control policies** to restrict the ability to **modify** the security attributes ***Card State*** to **the Issuer Security Domain, a Delegated Management Security Domain, the Card Lock Privileged Applets and the Card Terminate Privileged Applets**.

FMT_MSA.1-SD.4 Management of security attributes

FMT_MSA.1.1-SD.4 The TSF shall enforce the **Security Domain (SD) access control policies** to restrict the ability to **modify** the security attributes ***Closed Mode***, ***Package Properties*** to **the Issuer Security Domain**.

Application Note:

Depending on how the card was configured during its initialization, the Issuer Security Domain may either close the card automatically when it reaches GlobalPlatform's SECURED state, or upon reception of some APDU commands from the Card Administrator.

Package Properties may be modified using the SET PACKAGE PROPERTIES command.

FMT_MSA.3-SD Static attribute initialisation

FMT_MSA.3.1-SD The TSF shall enforce the **Security Domain access control policy (SD)** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2-SD The TSF shall allow the **Card Administrator** to specify alternative initial values to override the default values when an object or information is created.

Refinement:

- The *Default Selected* application shall be the ISD, by default.
- When the TOE enters the life cycle phases under the scope of this Security Target, the *Card State* shall be at least INITIALIZED.
- The initial value of the *Application State* of an applet instance shall be INSTALLED.
- The initial *Package Properties* shall enable all card content management operations on the package.

Application Note:

The Card Administrator may assign the Card Reset privilege and Implicit Selection Parameter to another application instance. The default value of the other security attributes of the SD policy cannot be modified.

FMT_SMR.1-SD Security roles

FMT_SMR.1.1-SD The TSF shall maintain the roles **Security Domain**.

FMT_SMR.1.2-SD The TSF shall be able to associate users with roles.

Application Note:

The term "security domain" is not used here with the technical meaning that GlobalPlatform attaches to it, but in the sense of an execution context separate from the one used for the applications instances, like the *JCRE execution context* defined in [JCRE].

FMT_SMF.1-SD Specification of Management Functions

FMT_SMF.1.1-SD The TSF shall be capable of performing the following management functions:

- **Setting the card to the Closed Mode, in which no further Executable Files can be loaded.**
- **Restricting the properties associated to a given package.**
- **Registering a new Executable File or application instance in the GlobalPlatform's registry.**
- **Removing the specified entries from the GlobalPlatform registry when a DELETE command is received.**
- **Setting an applet instance as the Default Selected Application.**
- **Granting the privileges when a new application instance is installed.**
- **Updating the privileges of an application instance.**
- **Extraditing an Executable File or application instance to another Security Domain.**
- **Moving the life cycle state of either an application instance or the whole card according to the life-cycle rules specified in [GPCS]**.

FMT_SMR.1-CA Security roles

FMT_SMR.1.1-CA The TSF shall maintain the roles **Card Administrator**.

FMT_SMR.1.2-CA The TSF shall be able to associate users with roles.

Application Note:

This SFR corresponds to FMT_SMR.1/CM of [JCSPP].

A Security Domain is just the on-card counterpart of a representative of the Card Issuer. It is introduced as a separate role in order to distinguish an application acting inside the smart card on behalf of the Card Issuer from the Card Administrator.

FTP_ITC.1-SC Inter-TSF trusted channel

FTP_ITC.1.1-SC The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2-SC The TSF shall permit **another trusted IT product** to initiate communication via the trusted channel.

FTP_ITC.1.3-SC The TSF shall initiate communication via the trusted channel for **all card content management operations:**

- **loading of Executable Load file;**
- **installation of application;**
- **extradition of Executable Load file or application instance;**
- **deletion of Executable Load file or application instance;**
- **GlobalPlatform registry update, including modification of Application Life Cycle or card Life Cycle;**
- **Personalization or replacement of SD key**.

Application Note:

This SFR corresponds to FTP_ITC.1/CM of [JCSPP].

There is no dynamic package loading on the Java Card platform. New packages can be installed on the card only on demand or upon authorization of the card issuer.

7.1.2 Card Management

This section describes the security functional requirements imposed on card management operations in addition to SFRs from the *CarG* group of requirements introduced in [JCSPP].

Card Management requirements are gathered into the following categories:

- Security Domain
- Secure Channels
- Card Content Management
- Global Cardholder Verification Method

Each category contains the security requirements concerning the implementation of a specific feature of GlobalPlatform.

7.1.2.1 Card Content Management

This section complements the security policy concerning the loading, installation and deletion of applications on the card. It also specifies requirements concerning the atomicity of those operations modifying the card contents, and the correct reception and interpretation of the code and the privileges of a new application.

FDP_ROL.1-CCM Basic rollback

FDP_ROL.1.1-CCM The TSF shall enforce **Security Domain access control policy (SD)** to permit the rollback of the **installation operation** on the **Executable Files and application instances**.

FDP_ROL.1.2-CCM The TSF shall permit operations to be rolled back within the **following boundary limit: until the Executable File or application instance has been added to the applet's registry**.

Application Note:

The platform shall be able to safely abort the loading of a new Executable File whenever a corrupted or out-of-sequence LOAD command is received, irrespectively to the length of the file being loaded.

FRU_RSA.1-CCM Maximum quotas

FRU_RSA.1.1-CCM The TSF shall enforce maximum quotas of the following resources: **imported Executable Files and declared classes, methods and fields** that **subjects** can use **simultaneously**.

Application Note:

The term "subjects" refer here to the Executable Files (or Packages, in Java Card jargon) that import or declare the listed resources.

FCO_NRR.1-RECEIPT Selective proof of receipt

FCO_NRR.1.1-RECEIPT The TSF shall be able to generate evidence of receipt for received **card content management operation requests** at the request of the **originator**.

FCO_NRR.1.2-RECEIPT The TSF shall be able to relate the **Confirmation Data (as defined in Section 11.1.6 of [GP])** of the recipient of the information, and the **parameters of the card content management operation request (as defined in Section C.5 of [GPCS])** of the information to which the evidence applies.

FCO_NRR.1.3-RECEIPT The TSF shall provide a capability to verify the evidence of receipt of information to **originator** given **successfull completion of the card content management request (load, install, extradition, registry update or delete)**.

7.1.2.2 Global Cardholder Verification Method

This section introduces the security requirements concerning GlobalPlatform's Cardholder Verification Method (CVM). This policy controls the modification of the internal data structures of the CVM that can be performed by the installed applications through the GlobalPlatform API.

FIA_AFL.1-CVM Authentication failure handling

FIA_AFL.1.1-CVM The TSF shall detect when **an administrator configurable positive integer within 1 and 255** unsuccessful authentication attempts occur related to **the authentication of the Cardholder**.

FIA_AFL.1.2-CVM [Editorially Refined] When the defined number of unsuccessful authentication attempts has been **met or surpassed**, the TSF shall **temporarily lock the Cardholder authentication service, until an unlocking action has been successfully undertaken by a privileged user**.

Application Note:

The PIN services created by the applications instances remain locked until the applet that owns the PIN object resets its state through the Java Card API.

7.1.2.3 Secure Channels

7.1.2.3.1 Key Loading Services

This section introduces the requirements for loading or replacing the static keys used to implement a Secure Channel. This mainly concerns the processing of the PUT KEY or STORE DATA commands sent to the SD.

FPT_TDC.1-KL Inter-TSF basic TSF data consistency

FPT_TDC.1.1-KL The TSF shall provide the capability to consistently interpret **the Security Domain keys** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2-KL The TSF shall use **the following rules:**

- **decryption of the data field of the received command shall be performed using the secure channel encryption session key**
- **before being set, imported AES and DES key values shall be decrypted using the data encryption session key**

when interpreting the TSF data from another trusted IT product.

FDP_ITC.1-KL Import of user data without security attributes

FDP_ITC.1.1-KL The TSF shall enforce the **Secure Channel Protocol (SCP)** when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.1.2-KL The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

FDP_ITC.1.3-KL The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **if the imported data is a key set for a SD, then:**

- **If the command specifies the replacement of a key set, the specified key set must exists on the card, it must have the same number of components than the proposed new one, and each component of the existing key must have the same type and length than the proposed new one.**
- **The type of the imported keys shall be supported by the configuration of GlobalPlatform specified in [GPCS] (DES128, AES128, RSA Public 1024, and EC Public 256 keys)**
- **The value of secret key components shall be decrypted using the DEK session key.**
- **If the loaded keys are DES or AES keys, then the presence of a key check value is required and its value shall be correct**.

7.1.2.3.2 Identification and Authentication

This section introduces the requirements concerning the actions that a subject may perform before opening a Secure Channel with the card.

FIA_UAU.1-SC Timing of authentication

FIA_UAU.1.1-SC The TSF shall allow **the TSF mediated actions listed in FIA_UID.1-SC** on behalf of the user to be performed before the user is authenticated.

FIA_UAU.1.2-SC The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

FIA_UAU.4-SC Single-use authentication mechanisms

FIA_UAU.4.1-SC The TSF shall prevent reuse of authentication data related to **the authentication mechanism used to open a secure communication channel with the card**.

7.1.2.3.3 SCP Information Flow Security Policy

This section introduces the requirements to be imposed on the Secure Channel linking the card and the card host when the channel is explicitly started. Those requirements come from the Secure Channel Protocols introduced in [GPCS].

FIA_AFL.1-SC Authentication failure handling

FIA_AFL.1.1-SC The TSF shall detect when **one** unsuccessful authentication attempts occur related to **the authentication of the origin of a card management command**.

FIA_AFL.1.2-SC [Editorially Refined] When the defined number of unsuccessful authentication attempts has been **met or surpassed**, the TSF shall **close the Secure Channel with the external user**.

FMT_MSA.1-SCP-OFFCARD Management of security attributes

FMT_MSA.1.1-SCP-OFFCARD The TSF shall enforce the **Secure Channel Protocol (SCP) information flow control policy** to restrict the ability to **modify** the security attributes ***Key Set***, ***Static Keys***, ***Command Security Level*** **of a Security Domain** to **Card Administrator (Security Domain)**.

Application Note:

This requirement specifies the security attributes that can be modified by the off-card users of which the Security Domain is the on-card representative. The value of those attributes is specified in the parameters of the commands that the off-card user sends to the Security Domain. The Key Set and the CAD Challenge are specified in the INITIALIZE-UPDATE command requesting the opening of a Secure Channel. The value of the Static Keys of the Security Domain is modified when the off-card user sends a PUT KEY command to the Security Domain. The Command Security Level is specified by the EXTERNAL-AUTHENTICATE command during the initialization of the Secure Channel. All those commands are actually processed only when the authentication step of the off-card user has been completed.

FMT_MSA.1-SCP-ONCARD Management of security attributes

FMT_MSA.1.1-SCP-ONCARD The TSF shall enforce the **Secure Channel Protocol (SCP) information flow control policy** to restrict the ability to **modify** the security attributes ***Session Keys*, *Sequence Counter* and *ICV*** to **the Security Domain**.

Application Note:

In the SCP02 protocol, the SD increments the *Sequence Counter* associated to the key set used to open the Secure Channel. The SD updates the *ICV* each time it successfully processes the cryptographic protections of an APDU command. All those attributes are reinitialized each time the off-card subject request the opening of a Secure Channel.

FMT_MSA.1-SCP-BOTH Management of security attributes

FMT_MSA.1.1-SCP-BOTH The TSF shall enforce the **Secure Channel Protocol (SCP) information flow control policy** to restrict the ability to **modify** the security attributes ***Cryptogram* of a Security Domain** to **the Security Domain itself and its off-card counterpart (Card Administrator)**.

Application Note:

The *Cryptogram* used by each subject to authenticate the other party is determined from the challenges exchanged by both subjects during the initialization of the Secure Channel.

FMT_SMF.1-SCP Specification of Management Functions

FMT_SMF.1.1-SCP The TSF shall be capable of performing the following management functions:

- **Generating a new card challenge during the set up of a Secure Channel.**
- **Generating the session keys for the Secure Channel from the specified static key set and its associated Sequence Counter.**
- **Generating the card cryptogram from the host and card challenges and the session keys.**
- **Increasing by one the Sequence Counter associated to the specified Key Set upon successful opening a Secure Channel.**
- **Setting the security level of the Secure Channel as the authenticated Card Administrator had specified during its set up.**
- **Updating the current value of the ICV upon reception of a new message through the Secure Channel.**
- **On request of a Card Administrator, loading or replacing the static keys that the associated Security Domain uses to open a Secure Channel**
- **Management functions (concerning Secure Channel) specified in GlobalPlatform specification [GPCS]:**
  - **store personalization data**
  - **key loading**.

FMT_MSA.3-SCP Static attribute initialisation

FMT_MSA.3.1-SCP The TSF shall enforce the **Secure Channel Protocol (SCP) information flow control policy and the PACKAGE LOADING information flow control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2-SCP The TSF shall allow the **Card Administrator** to specify alternative initial values to override the default values when an object or information is created.

Refinement:

The Card Administrator may specify alternative default values for the *Static Keys* attribute of the SCP policy.

Application Note:

This SFR corresponds to FMT_MSA.3/CM of [JCSPP].

In [JCSPP], no role is authorized to provide alternative default values for the security attributes of the information flow control policy of the *Carrier* group. Such assignment seems unnecessary: as the protection profile does not specify which the security attributes of the policy are, stating that they cannot have alternative values seems to be an over-specification. For instance, such assignment is in conflict with GlobalPlatform's specifications. For this reason, this Security Target does not follow the Java Card Protection Profile on this point, and allows the off-card counterpart of a Security Domain to specify alternative default values for the static keys associated to a Secure Channel.

7.1.2.3.4 Key Generation

This section specifies the cryptographic requirements concerning the generation and distribution of the session keys used for setting up the Secure Channel Protocol.

FCS_CKM.1-SCP-SESSION-KEYS Cryptographic key generation

FCS_CKM.1.1-SCP-SESSION-KEYS The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **(the session key generation algorithm described in Appendix E.4.1 of [GPCS])** and specified cryptographic key sizes **(16-bytes key)** that meet the following: **ANSI X9.52 and ISO 10116**.

Application Note:

The algorithm used to generate the session key includes the encryption of a derivation data depending on the session using a Triple DES in CBC mode. Such encryption algorithm shall follow the standards cited in this component.

FCS_CKM.2-SCP-SESSION-KEYS Cryptographic key distribution

FCS_CKM.2.1-SCP-SESSION-KEYS The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **(the SCP02 cryptographic protocol)** that meets the following: **[GPCS]**.

Application Note:

During the set up of a Secure Channel, the TOE agrees with the card host on the session key to be used for that channel. The TOE never distributes the session key (in the sense of sending the whole key to the host) but it exchanges information with the host that enable each party to derive the session key on its own side. This requirement comes form [JCSPP]. It has been placed in the Card Management section for the sake of clearness.

7.1.2.3.5 Cryptography Operations

This section introduces the cryptography requirements concerning the verification of the origin, integrity and confidentiality of the card management commands received through a Secure Channel.

FCS_COP.1-GP Cryptographic operation

FCS_COP.1.1-GP The TSF shall perform **[assignment: list of cryptographic operations]** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithm]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]:**

Iteration	Operation	Algorithm	Key Size	Standard
-SCP02/CBC	session key derivation and data field decryption of the messages exchanged through GlobalPlatform's Secure Channels	Triple DES in CBC mode	112 bits	FIPS PUB 46-3, ANSI X9.52 and ISO 10116.
-SCP02/ECB	key encryption/decryption and DES key's check value generation	Triple DES in ECB mode	112 bits	FIPS PUB 46-3, ANSI X9.52 and ISO 10116
-SCP02/ICV	encryption and decryption of message integrity ICV	Simple DES in ECB mode	56 bits	FIPS 46-2
-SCP02/FINAL	MAC verification of the messages exchanged through a secure channel	single DES plus final Triple DES	128 bits	FIPS PUB 46-3, ISO 9797-1

Iteration	Operation	Algorithm	Key Size	Standard
-SCP/FULL	authentication cryptogram generation and verification and MAC verification of the messages exchanged through a secure channel	full triple DES	128 bits	FIPS PUB 46-3, ISO 9797-1
-DAP	verification of the DAP signature attached to Executable Load Files	RSA SSA-PKCS1-v1.5 using SHA-1 digest, AES128 NIST SP 800-38B, ECDSA 256	1024 bits for RSA, 128 bits for AES and 256 bits for ECDSA	PKCS#1 (RSA), NIST SP 800-38B (AES), BSI TR 03111 (ECDSA)
-DM/TOKEN	verification of the Delegated Management Token signature attached to card content management commands	RSA SSA-PKCS1-v1.5 using SHA-1 digest, AES128 NIST SP 800-38B, ECDSA 256	1024 bits for RSA, 128 bits for AES and 256 bits for ECDSA	PKCS#1 (RSA), NIST SP 800-38B (AES), BSI TR 03111 (ECDSA)
-DM/RECEIPT	generation of the Delegated Management Receipt signature attached to responses to card content management commands	DES signature, AES signature	128 bits for DES and 128 bits for AES	ISO 9797 M2 ALG3 (DES), NIST SP 800-38B (AES)

.

7.1.2.4 Security Domain

This section introduces an access control policy containing the requirements that are specific to the card management operations that GlobalPlatform provides: enforcing the life cycle of the card and its applets, loading and installing new applets, removing existing ones, loading new cryptographic keys, and setting and retrieving information about the card's history. This policy enforces the policy that GlobalPlatform's specifications assigns to that distinguished application. That policy does not only concern card content management, but also other aspects of card management activities, like controlling the life cycle of the card and the applets.

FDP_ACC.1-SD Subset access control

FDP_ACC.1.1-SD The TSF shall enforce the **Security Domain (SD) access control policy** on **the following list of subjects, objects and operations:**

- **The subjects controlled by the policy are the applications instances installed on the card. Among those applications there is are distinguished ones, called Security Domains.**
- **The objects controlled by the policy are:**
  - **The Executable Files loaded on the platform**
  - **The Executable Modules defined in the Executable Files**
  - **The application instances created on the card**
  - **The keys handled by the security domain**
  - **The personalization data for both the card and the applications**
- **The operations controlled by the policy are GlobalPlatform's APDU commands and API methods**.

Application Note:

GlobalPlatform specifies the following APDU commands:

- DELETE: deleting an Application Instance or an Executable File
- EXTERNAL AUTHENTICATE: authenticating the host and determining the security level of the secure channel
- GET DATA: retrieving card administration information
- GET STATUS: retrieving loaded keys, installed Executable Files and Application instances, and their current life cycles
- INITIALIZE UPDATE: requesting the opening of a secure channel
- INSTALL[for install and make selectable]: installing a new application instance
- INSTALL[for load]: installing a new Executable Load File
- INSTALL[for extradition]: extradition of an Application or an Executable Load File from a Security Domain.
- INSTALL[for personalization]: start a personalization session
- INSTALL[for registry update]: preventing loading any further Executable File on the card
- SET PACKAGE PROPERTIES: restricting card content actions on a given Executable File
- GET PACKAGE PROPERTIES: listing authorized card content actions on a given Executable File
- LOAD: loading of a piece of a Executable Load File
- PUT KEY: loading or replacing a key set
- STORE DATA: transfer data to an application or the Security Domain processing the command
- SELECT: selecting of an application instance for execution
- SET STATUS: modifying the life cycle state of either the card or an application

The SD access control policy also controls the use of the following methods of the GlobalPlatform API:

FDP_ACF.1-SD Security attribute based access control

FDP_ACF.1.1-SD The TSF shall enforce the **Security Domain access control policy (SD)** to objects based on the following:

All application instances, including the SD, have the following security attributes:

- **The *Closed Mode* attribute specifies whether the TOE has been configured to prevent the loading of additional Executable Files.**
- **The *Card Reset* and *Implicit Selection* attributes specify whether the applet instance is the one that should be executed when no application has been explicitly selected on the logical channel used.**
- **The *Application State* attribute specifies the current life cycle state of the application instance, which may be either SELECTABLE, APPLICATION_SPECIFIC,LOCKED.**
- **The *Card Lock* attribute specifies whether the applet is allowed to temporary lock the services of the smart card**
- **The *Card Terminate* attribute specifies whether the applet is allowed to definitely disable the services of the smart card.**
- **The *CVM* attribute specifies whether the applet is allowed to modify the try limit and the PIN code of the global CVM service. In addition to the ones above, the SD has the following extra security attributes:**
- **The *CardState* attribute, is the current state in the life cycle of the card, which may be either OP_READY, INITIALIZED, SECURED, CARD_LOCKED, or TERMINATED.**
- **The *Registered Applications* attribute specifies the Executable Files and application instances that have been installed on the card so far and their dependencies.**
- **The DAP Verification checks integrity and authenticity of an application code to be loadded on the card.**
- **The Mandated DAP Verification checks integrity and authenticity of all applications code to be loadded on the card.**
- **The Delegated Management allows an Application Provider to manage Card Content with authorization.**
- **The Authorized Management allows the ISD to manage Card Content without authorization.**
- **The Global Registry specifies wether the applet can access any entry in the GlobalPlatform Registry.**
- **The Global Lock specifies whether the applet is allowed to lock or unlock any application.**
- **Implicit Selection Parameters indicates whether an application is implicitly selectable on a specific logical channel.**

The packages under the control of this security policy have the following attributes:

- **The *Package Properties* attribute specify whether the following restrictions apply to the applet classes that the package declares: selection of applet instances in contact-based mode is disabled.**
  - **selection of applet instances in contactless mode is disabled.**

- **installation of additionnal instances of those classes is disabled.**
- **installation of additional instances is disabled, provided that one single instance has been created at the moment of the installation request.**
- **deletion of any instance of those classes is disabled**.

FDP_ACF.1.2-SD The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **Rule SD-1: A card administration request may be accepted only if the APDU command specifying the request is well-formed according to [GPCS].**
- **Rule SD-2: A card administration request other than requesting card management data may be accepted only if the *Card State* is not TERMINATED.**
- **Rule SD-3: The selection of an applet instance different from a SD may be accepted only if the properties of the package declaring the applet enable selection through the interface used to send the command, and the *Applet State* is not LOCKED.**
- **Rule SD-4: The update of the life cycle state of an application instance is accepted only if the new state is consistent with its current life cycle state according to GlobalPlatform's life cycle rules (either coming from an APDU command or from an application instance through the GP API).**
- **Rule SD-5: A request for installing an Executable Load File may be accepted only if the card has not been set to the Closed Mode, there is enough resources for loading the Executable File, and no Executable File on the card has been already registered with the specified AID.**
- **Rule SD-6: An Executable Load File block may be loaded only if the card has not been set to the Closed Mode, all its previous blocks have been received in order, and there are sufficient resources for storing the new one. If the mandated DAP Verification is required, the block must be checked for integrity and authenticity.**
- **Rule SD-7: A new applet instance may be created; the Package Properties allow applet instantiation and (if an instance of that applet is already installed on the card) multiple applet instances, the AID specified for the applet instance is not already used for another applet or Executable File installed on the card, and the privileges specified for it are consistent with the configuration of GlobalPlatform specified in [GPCS/ID].**
- **Rule SD-8: An Executable File may be deleted from the smart card only if it is not reachable from other Executable Files or application instances on the card, it is not necessary for the working of the platform, like for instance, the Java Card or GlobalPlatform APIs. The associated SD shall accept deletion if the delete request comes from a SD with DM privileges or the ISD.**
- **Rule SD-9: An applet instance may be deleted from the card only if its Package Properties enable deletion, the instance is not the ISD, it is not currently active on a logical channel, and none of the Java Card objects it has allocated is reachable from other Executable Files or Application**

instances installed on the card. The associated SD shall accept deletion if the delete request comes from a SD with DM privileges or the ISD.

- **Rule SD-10: An applet instance may lock the card only if it has the *Card Lock* privilege.**
- **Rule SD-11: An applet instance may terminate the card only if it has the *Card Terminate* privilege.**
- **Rule SD-12: An applet instance may unlock the CVM service or modify the CVM try limit or PIN code only if it has the *CVM* privilege.**
- **Rule SD-13: A request involving the use of any of the SD keys is accepted only if the concerned keys are integer with respect to their associated checksum values.**
- **Rule SD-14: A SSD with Delegated Management (DM) privilege must provide a successful Token Verification**.

FDP_ACF.1.3-SD The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4-SD The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **following rule: when at least one of the conditions listed in the element FDP_ACF.1.2 of this component does not hold**.

FMT_SMR.1-PRV Security roles

FMT_SMR.1.1-PRV The TSF shall maintain the roles **Default Selected Applet, Card Lock Privileged Applet, Card Terminate Privileged Applet, CVM Privileged Applet, DAP Verification Privilege, Mandated DAP Verification Privilege, Delegated Management Privilege, Authorized Management Privilege, Global Registry Privilege, Global Lock Privilege and Implicit Selection Install Parameters**.

FMT_SMR.1.2-PRV The TSF shall be able to associate users with roles.

7.1.3 Runtime Environment

This section contains the security functional requirements concerning the Runtime Environment that executes the applets. The requirements are gathered into the following categories:

- Core Requirements, which corresponds to the *CoreG* group of requirements specified in [JCSPP].
- Applet Deletion, which corresponds to the *ADEL* group of requirements specified in [JCSPP].
- Garbage Collection, which correspond to the *ODEL* group of requirements specified in [JCSPP].
- Defensive Virtual Machine requirements, which specify an additional access control policy that jTOP's virtual machine enforces at runtime.

- Operating System Requirements, which corresponds to a subset of the *Secure Card Platform* group of requirements in [JCSPP], namely, those concerning the implementation of the Operating System of the card.

7.1.3.1 Defensive Virtual Machine

This group of requirements concerns the dynamic verifications of the applet's bytecode that jTOP's Java Card Virtual Machine implements. Some of these verifications are redundant with respect to the BCVG group of security requirements that [JCSPP] imposes on the IT environment. They constitute a second line of verifications for counter those attacks based on the execution of ill-typed applications.

FDP_ACC.1-DVM Subset access control

FDP_ACC.1.1-DVM The TSF shall enforce the **Defensive Virtual Machine (DVM) access control policy** on **the following list of subjects, objects and operations:**

- **The subjects under the control of the DVM security policy are the Application instances installed on the card.**
- **The objects under the control of the DVM security policy are:**
  - **the Java Card class instances and arrays**
  - **the code of the Executable Files**
  - **the static data of the Executable Files (static field image)**
  - **the runtime data areas of the JCVM: the operand stack and local variables of the currently executed method**
- **The operations under the control of the DVM security policy are the Java Card bytecodes**.

FDP_ACF.1-DVM Security attribute based access control

FDP_ACF.1.1-DVM The TSF shall enforce the **Defensive Virtual Machine (DVM) access control policy** to objects based on the following:

- **The following security attribute is associated to the operand stack:**
  - ***Stack Bounds*: the bounds of the memory zone reserved to hold the operand stack of the JCVM;**
- **The following security attribute is associated to the local variables of the method:**
  - ***LocVar Bounds*: the bounds of the memory zone reserved to hold the local variables of the current method;**
- **The following security attribute are associated to dynamically allocated objects, like class instances, arrays and downloaded Executable Files:**
  - ***Bounds*: the bounds of the memory zone containing the dynamically allocated object;**
  - ***Allocated*: whether the memory zone containing the object was previously allocated by the virtual machine**

- ***Use*: whether the memory zone containing the object was allocated for storing a piece of code or a piece of data**.

FDP_ACF.1.2-DVM The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **Rule DVM-1: An applet may pop values from the operand stack only if there are enough values on the stack to perform such operation.**
- **Rule DVM-2: An applet may push values onto the operand stack only if there is enough room on the operand stack for the new values.**
- **Rule DVM-3: An applet may read or write a local variable of a given method only if the local variables are within the ones that the applet declared for this method.**

[information removed]

FDP_ACF.1.3-DVM The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4-DVM The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **following rule: when none of the rules listed in the element FDP_ACF.1.1 of this component is satisfied**.

FMT_MSA.3-DVM Static attribute initialisation

FMT_MSA.3.1-DVM The TSF shall enforce the **Defensive Virtual Machine (DVM) access control policy** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2-DVM The TSF shall allow the **following roles: none of them** to specify alternative initial values to override the default values when an object or information is created.

7.1.3.2 Core Requirements

This group is focused on the main security policy of the Java Card System, known as the Firewall. This policy essentially concerns the security of installed applets.

7.1.3.2.1 Card Security Management

The following requirements are related to the security of the whole card, in contrast to the previous ones, that are somewhat restricted to the features of the Runtime Environment alone. For instance, a potential security violation detected by the virtual machine may require a reaction that does not only concern the virtual machine, such as requesting the appropriate security module with the power to block the card to perform the operation.

FPT_TST.1 TSF testing

FPT_TST.1.1 The TSF shall run a suite of self tests **during initial start-up** to demonstrate the correct operation of **the TSF depending on specific support from the IC**.

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of **TSF data**.

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of **stored TSF executable code**.

Refinement:

The initial start-up corresponds to power on.

7.1.3.2.2 Smart Card Platform

The *Smart Card Platform* group introduced in [JCSPP] specifies the IT requirements that are imposed on the Operating System and the Integrated Circuit underlying the implementation of the Runtime Environment. Because of the modification in the scope of evaluation, which does include in this Security Target the Operating System and the Integrated Circuit, those requirements on the IT environment become requirements on the TOE itself.

FPT_RCV.3-OS Automated recovery without undue loss

FPT_RCV.3.1-OS When automated recovery from **security policy violation** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2-OS For **execution access to a memory zone reserved for TSF data, writing access to a memory zone reserved for TSF's code, and any segmentation fault performed by a Java Card applet**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3-OS The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding

- **the contents of Java Card static fields, instance fields, and array positions that fall under the scope of an open transaction;**
- **the Java Card objects that were allocated into the scope of an open transaction;**
- **the contents of Java Card transient objects;**
- **any possible Executable Load File being loaded when the failure occurred**

for loss of TSF data or objects under the control of the TSF.

FPT_RCV.3.4-OS The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.

FPT_RCV.4-OS Function recovery

FPT_RCV.4.1-OS The TSF shall ensure that **reading from and writing to static and objects fields interrupted by power loss** have the property that the function either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

7.1.3.3 Miscellaneous

FMT_MSA.2-KEYS Secure security attributes

FMT_MSA.2.1-KEYS The TSF shall ensure that only secure values are accepted for **all the TOE security attributes**.

Application Note:

Before using a key, the platform shall check that the operation to be performed with it is in accordance with its secure attributes, like its length, associated algorithm (DES, RSA, etc) and key type (public, secret).

FIA_AFL.1-KEYS Authentication failure handling

FIA_AFL.1.1-KEYS The TSF shall detect when **an administrator configurable positive integer within 1 and 255** unsuccessful authentication attempts occur related to **the verification of a cryptographic signature associated to a given key**.

FIA_AFL.1.2-KEYS [Editorially Refined] When the defined number of unsuccessful authentication attempts has been **met or surpassed**, the TSF shall **multiply the response time of the next signature verification operation by an administrator configurable factor**.

Application Note:

This mechanism aims to prevent brute force attacks. It can be activated during the card initialization phase. Once activated, it applies to all the signature verification services provided by the card, including those services that applets create through the Java Card API.

7.2 Security Assurance Requirements

The Evaluation Assurance Level is EAL5 augmented with ALC_DVS.2 and AVA_VAN.5.

8 TOE Summary Specification

8.1 TOE Summary Specification

This section describes the security functionalities of the TOE.

The security functionalities concerning the IC are described in [ICST] and are not redefined in this security target, although they must be considered for the TOE. The security functionalities from [ICST] are reminded below:

- SF_DPM Device Phase Management
- SF_PS Protection against Snooping
- SF_PMA Protection against Modification Attacks
- SF_PLA Protection against Logical Attacks
- SF_CS Cryptographic Support

8.1.1 Card Management

This section introduces the security functionalities that the GlobalPlatform layer provides.

8.1.1.1 Security Domain

The following security functionalities concern the security features of the Security Domain.

OPEN

This TSF is in charge of selecting the applet instances to be activated (selected for execution), and dispatching the received commands to them. It also initializes and manages the internal data structures required for implementing the card management services. In GlobalPlatform, this functionality is referred to as the OPEN (Open Platform ENvironment).

[information removed]

Card Content Management

This TSF controls the loading, installation, and removal of Executable Files and application instances.

[information removed]

Life Cycle Management

This TSF enforces the life cycle that GlobalPlatform defines for both the card and the installed applications in [GPCS]. It also controls that the card cannot shift from a GlobalPlatform state to one of the proprietary states of the Card Initialization phase.

[information removed]

Administration Commands Control

This TSF checks that only the card management commands specified in [FSP] are accepted, and rejects ill-formed ones with an appropriate error response. It also controls which card administration commands are allowed at each state of the smart card's life cycle.

[information removed]

8.1.1.2 Secure Channels

Most of card management duties involve importing or modifying security sensitive assets, like cryptographic keys or platform personalization data. The TOE uses the Secure Channel Protocol 02 (SCP02) specified in [GPCS] to protect such assets while they are in transit to the card, and to reject any piece of data coming from a distrusted user. This cryptographic protocol consists of three main steps: mutual authentication, message exchange and secure channel termination.

Host Authentication

This TSF enforces the authentication of the Card Administrator through the Secure Channel Protocol 02 (SCP02).

[information removed]

Message Integrity and Authentication

This TSF enforces the integrity and the origin of the APDU commands received through a Secure Channel.

[information removed]

Message Data Confidentiality

This TSF decrypts the contents of an APDU message containing sensitive information.

[information removed]

Secure Channel Termination

When a Secure Channel is closed this TSF ensures that the session keys are cleared and the security level is reset. Further commands received on the underlying logical channel are not considered as coming from the Card Administrator.

[information removed]

8.1.1.3 Secure Channel Key Management

The following security functionalities concern the keys required to set up a secure communication channels between the smart card and its host.

8.1.1.3.1 Integrity

The following security functionalities contribute to enforce data integrity requirements.

Atomic Transactions

This TSF provides means to execute a sequence of modifications and allocations on the persistent memory so that either all of them are completed, or the card behaves as if none of them had been executed. The transaction mechanism is used for updating internal sensitive data as well as for performing different card management functionalities, such as installing a new Executable File on the card or creating a new applet instance.

[information removed]

8.1.1.3.2 Miscellaneous

Session Key Generation

This TSF controls the generation of the session keys used to set up a Secure Channel with the CAD.

[information removed]

SD Key Loading and Replacement

This TSF enables to load or replace one of the key sets that the SD uses to establish a secure channel with the Card Administrator.

[information removed]

8.1.1.4 Cardholder Verification Management

The following security functionality concerns the management of a global service for authenticating the Cardholder.

Global CVM

This TSF controls the update of the security attributes associated to a global cardholder authentication service shared by all the applications installed on the platform.

[information removed]

8.1.2 Runtime Environment

This section introduces the security functionalities that the Runtime Environment provides.

8.1.2.1 Application Reference Monitors

The following security functionalities provide abstract machines that mediate the access to objects by subjects during the interpretation of the applet's bytecode.

Java Card Firewall

This TSF enforces the Firewall access control policy and the JCVM information flow control policy at runtime. The former policy controls object sharing between different applet instances, and between applet instances and the Java Card Runtime Environment. The latter policy controls the access to global data containers shared by all applet instances.

[information removed]

Defensive Java Card Virtual Machine

This TSF is a reference monitor on the actions that the application instances perform at runtime.

[information removed]

8.1.2.2 Security Countermeasures

The following security functionalities concern the countermeasures that the TOE may trigger upon detection of a security policy violation.

Card Muting

This TSF makes the card to stop processing the current command and shifts it into a state where any further communication with it is impossible from outside until the next card reset (cold or warm).

When the card is muted, it does not react to any APDU command and sends no more answers to the CAD.

Card Locking

This TSF moves the card to a life cycle state where all applications except the Security Domain are disabled and cannot be selected for execution.

When the card is locked, its services are reduced to those offered to the Card Administrator, which require the authentication of this actor. In addition to this, the SD rejects loading, installing or deleting any Executable File or Application instance until the card is unlocked.

[information removed]

Card Termination

This TSF moves the card to a state where all the servicesoffered by the smart card are definitely disabled, except for card management information retrieval. Any other information stored in the smart card such as keys or PIN codes is cleared.

When the card is terminated, applet selection is impossible and the card rejects any APDU command except GET DATA.

[information removed]

8.1.2.3 Life Cycle Management

The following security functionalities concern specific security actions that are performed upon special events in the life cycle of the platform.

Clearing Sensitive Information

This TSF clears all the data containers that hold sensitive information when that information is not longer used.

[information removed]

Booting Tests

This TSF checks at the beginning of each card session the correct initialization of all the security services, the integrity of the TSF data on which they relay. It also prevents the platform code from running on a chip different from the one for which it was designed.

[information removed]

8.1.2.4 Service Availability

The following functionalities contribute to prevent denial of service.

Resource Quotas

This TSF ensures that each application instance meets the memory quotas that the Card Administrator assigned to it when the instance was installed on the card. When the memory quota allocated for an application instance is exhausted, this TSF checks that no further memory is allocated to it. A maximum amount of memory may be also defined for all the instances of the classes declared in a given Executable File.

This TSF also checks that the Executable Files installed on the card comply with the CAP format constraints regarding the number of imported packages and the number of declared classes, instance fields and static fields that an application may define, according to [JCVM].

8.1.2.5 Cryptography

The following security functionalities implement the cryptography algorithms required for the platform.

Signature Generation and Verification

This TSF provides applets with a mechanism for generating an electronic signature of a byte array, and verifying an electronic signature stored in a byte array.

Signature algorithms are available to the applications through the `javacard.security.Signature` abstract class of the Java Card API, for which the Runtime Environment provides a concrete implementation.

Java Card Technology enables a given platform to implement a variable set of signature algorithms. The signature key types and algorithms from the Java Card API that the TOE supports are summarized in the reference [PROFILE].

If the applet specifies an algorithm that the platform does not support, the Runtime Environment refuses to create the requested signature instance. Some other signature algorithms, even though supported, do not fall into the scope of the TOE, and their use is not advised; see [USR].

[information removed]

Encryption and Decryption

This TSF provides the applets with a mechanism for encrypting and decrypting the contents of a byte array.

Ciphering algorithms are available to the applications through the `javacardx.Cipher` abstract class of the Java Card API, for which the Runtime Environment provides a concrete implementation.

Java Card Technology enables a given platform to implement a variable set of cipher algorithms. The key types and encryption algorithms from the Java Card API that the TOE supports are summarized in the reference [PROFILE].

[information removed]

Message Digest Generation

This TSF provides the application instances with a mechanism for generating an (almost) unique value for the contents of a byte array. Such a value can be used as a shorter available of the information contained in the whole byte array.

Message Digest algorithms are available to the applications through the `javacard.security.MessageDigest` abstract class of the Java Card API, for which the Runtime Environment provides a concrete implementation. The message digest algorithms from the Java Card API that the TOE supports are summarized in the reference [PROFILE].

[information removed]

Random Number Generation

This TSF provides the application instances with a mechanism for generating a randomly chosen piece of data. Such data is intended to be used for generating protocol challenges and key values.

Random number generators are available to the applets through the `RandomData` class of the Java Card API. The random numbers provided to the applets are compliant with DCSSI's "STANDARD" quality metric specified in [ANSSI].

[information removed]

8.1.2.6 Key Management

The following security functionalities implement key infrastructure requirements.

Key Generation

This TSF provides the applets with on-card generation of cryptographic keys.

It only supports generation of RSA and EC key pairs. Key components are generated using a secure random number generator compliant with DCSSI's *Standard* security level for cryptography operations.

[information removed]

Key Agreement

This TSF enables an applet to agree on a shared secret with the terminal, without disclosing this secret to third parties that could observe the messages exchanged between the card and the terminal.

[information removed]

Key Encryption

This TSF protects the cryptographic keys from being read out from the memory. It ensures that keys are stored into secure containers in an encrypted format and provides the functionalities for accessing and modifying them in a secure container.

[information removed]

Key Integrity

This TSF checks the integrity of the keys before performing any cryptographic operation with them.

[information removed]

Key Destruction

This TSF disables the use of a key both logically and physically.

[information removed]

8.1.2.7 Cardholder Authentication

The following functionalities implement the requirements concerning the authentication of the Cardholder.

Cardholder Verification

This TSF enables applet instances to authenticate the sender of a request as the genuine Cardholder. Applets have access to this service through the `OwnerPIN` class of the Java Card API.

[information removed]

PIN Value Integrity

This TSF checks the integrity of the PIN and its persistent attributes (try-counter, try-limit and CVM state) before each Cardholder authentication attempt.

[information removed]

Index